Fundamentals of modern photo-composition

John W. Seybold

Fundamentals Of Modern Photo-Composition

John W. Seybold

Seybold Publications, Inc. • Box 644 • Media, Pennsylvania 19063

© 1979 by Seybold Publications, Inc.
Box 644, Media, Pennsylvania 19063

All rights reserved. No part of this book may be reproduced in any form or by any means without permission in writing from the publisher.

10 9 8 7 6 5 4 3 2 1

Printed in the United States of America

ISBN: 0-918514-02-8 (paper)
0-918514-03-6 (cloth)

To JOHN J. TUOHEY

Vice President of Manufacturing
U.S. News & World Report
Whose unusual understanding
of practical publishing economics
and present-day technology,
coupled with his insights
into the future,
have provided me with rare experiences
as I have looked over his shoulder
and worked on his team.

To Stephen Edwards and Martha Morrison

Whose patience, persistence
and dedication to this project—
inputting, proofreading, performing
page make-up, and handling all of
the complex production details—
have made this new edition
come to pass.

And to Trudie

Whose tolerance of the idiosyncrasies
of an overcommitted writer and consultant
and whose forebearance and good humor
passeth even all wifely understanding.

This book is dedicated.

J.W.S.

Table of Contents

	Introduction	i
CHAPTER ONE	How Type Composition Differs from Typewriter Composition	1
CHAPTER TWO	Further Comparisons between Typewriter and Typesetter	8
CHAPTER THREE	Characteristics of Type—Basic Concepts	17
CHAPTER FOUR	The Linecasting Machine and Its Relation to the Production Process	34
CHAPTER FIVE	Monotype Composition: The Process and Its Contribution	45
CHAPTER SIX	The Advent of TTS	58
CHAPTER SEVEN	The Beginnings of Photocomposition (First-Generation Machines)	72
CHAPTER EIGHT	Underlying Principles of Second-Generation Typesetters	87
CHAPTER NINE	Third-Generation (CRT) Typesetters and Subsequent Technologies	119
CHAPTER TEN	Basic Computer Concepts	157
CHAPTER ELEVEN	Programs and Programming	180
CHAPTER TWELVE	Some Introductory Thoughts about Typesetting "Systems"	189
CHAPTER THIRTEEN	The Composition Process	197
CHAPTER FOURTEEN	Hyphenation Examined	219
CHAPTER FIFTEEN	Editing and Corrections	234
CHAPTER SIXTEEN	Input into a Typesetting System	247
CHAPTER SEVENTEEN	Optical Character Recognition	262
CHAPTER EIGHTEEN	Video Terminals and Video Editing Systems	287
CHAPTER NINETEEN	Disc Systems and File Management Schemes	310
CHAPTER TWENTY	Word Processing and Its Relationship to the Composition Process	319
CHAPTER TWENTY-ONE	Pagination by Computer and "Area Composition"	328
CHAPTER TWENTY-TWO	The Generation of Halftones	347
CHAPTER TWENTY-THREE	The Typesetting/Data Processing Relationship	368
CHAPTER TWENTY-FOUR	Putting the System Together	385
	Index	
	Alphabetical List of Illustrations	

INTRODUCTION

THIS BOOK is written for what we hope will be a broad public: for writers, for editors, for traditional typesetters and printers, for entrepreneurs, for publishers contemplating the installation of a "system" for editorial "input," for teachers and students of journalism and printing, and, of course, for purchasers of composition who will need to know what alternatives are now available to them. In order to address ourselves to such an audience—all concerned in one fashion or another with the same problem, but diverse in their backgrounds, experiences, and especially in their understanding of one another's tasks and duties, we shall have to take very little for granted. It will be necessary for us to proceed—at least at times—in a fairly elementary fashion, to build the foundation upon which an appreciative comprehension of the evolving technology may be developed.

Our task is not an easy one, because we lack the vocabulary necessary to communicate with one another. There is no one term, for example, which embraces the area of our interest and is truly descriptive of our subject matter. Some people say that our concern is with "pre-press" activity, but this term is at once too narrow and too broad. It is too narrow because "pre-press" functions are not normally understood to embrace writing and editorial tasks. It is too broad a term since there are other operations, having to do with imposition, and the processing of film, and the making of printing plates, which are only incidentally within the scope of this book—although each day their relevance to our area of concern becomes more evident. Our attention will be focussed particularly upon the editorial/typesetting interface. *Interface* means a relationship between—the manner in which two independent entities transact business—*i.e.,* communicate—with each other. Obviously, the purpose of a book such as this is to strengthen and deepen all such interfaces, including the relationships between technical people and creative people, typesetting craftsmen and data processing experts, as well as scientists and artists.

The phrase *word processing* might be used to describe the kind of activity we are in part concerned with, since it is a term which implies some participation by the writer himself and, by inference, embraces all subsequent operations up to the point of plate-making and/or printing. But that term has, perhaps unfortunately, been pre-empted to describe certain activities which are less germane to printing and publishing than they are to the typing of letters and documents, proposals and forms, briefs and specifications; and these "office-related" functions are not necessarily carried forward with the thought of publication in mind. Nevertheless, we shall find that the needs and the tools of the so-called "word processing" industry are similar in many respects to those we shall be writing about, and, indeed, we have prepared a separate chapter which is concerned with the interrelationships between "word processing" and publishing.

As you can see, we have more success in explaining what our area of interest is *not* than what it truly is. We shall let the matter rest and hope that the reader will discover his own term for the subject at hand, and perhaps the boundaries which that term embraces.

As with most areas of technology, it is often many years before it is possible to explain what it is that we "have," what it means or portends, and how it can be used. New disciplines are involved: the practitioners lead the way, the scholars follow afterward and try to provide a coherent account of what has transpired and why. In the meantime, we shall have to suffer with some degree of imprecision, and there will be some fuzziness when it comes to stating explicitly the beginning and ending of our inquiry. The fact is that at this point we simply do not know where all of this is leading.

The impact upon the quality of writing

FOR EXAMPLE, one speculation that we have decided not to entertain has to do with the possible impact of new writing and editing technology upon the nature—*and especially the quality*—of creative writing. On one hand it may be contended that if the process of recording one's thoughts becomes much more simple, thus facilitating spontaneity, the written product will be more slapdash in character—more poorly organized and perhaps less cogently phrased—because the author will not have had to *labor* so much at his writing task. On the other hand, it has been suggested that the facility with which changes can be made—massive deletions accomplished and entire paragraphs moved from place to place—should make it possible for the writer to do a better and more coherent job. It has been argued that the typewriter itself has not promoted great writing and that authors should be obliged to scratch out their immortal lines with a quill pen. Some critics have implied that we would all be better off if what was to be written should be tooled by the author into marble so that he would not be encouraged to run off at the mouth and would choose to pass on to posterity only the *essence* of his wisdom and inspiration.

It may be true that the *means* available to the author do affect—for better or for worse—the quality and quantity of his work. But writing habits are so highly individualized that it would be dangerous, if not absurd, to hazard any generalizations about the impact of these new tools of the trade upon the subject matter itself. All that we can say for certain is that since the time interval between the conception of the idea and its appearance in print can now—under many circumstances—be dramatically reduced, what is written may be more pertinent to our times and more useful to help shape the future course of history.

No, we shall refrain from any speculation about the effect upon the writer's product, although the subject intrigues us. But we cannot so readily ignore the concern that some editors and writers have expressed that they may well have less time to do their *proper* job if a large proportion of their day has to be devoted to the creation of *input* that is automatically fed into *the system*, by means of the *capture* of their *keystrokes*. This concern will cause us to examine the extent to which the author (as for example, reporter), or the editor, is in fact inhibited, or whether he/she may not be "liberated" by the new technology. The burden of accuracy and stylistic consistency may not be so onerous for the future as it has been in the past since there are certain kinds of functions of an editorial nature which computers can perform unobtrusively, and some publishers, or even some editors, may come to appreciate the assistance which can be so rendered.

But we are obviously flirting with the intriguing peripheral aspects of our topic before we have come to grips with its substance. We have injected considerations which relate to the *effects* of the new technology upon the people and upon the *quality of their endeavors* in order to underscore two caveats which we should be careful to remember. One is that we must not be overwhelmed by the technology itself, simply because it is

there. And second, as a corollary, *people* should be the masters, and while technology should be the servant, the opportunities afforded by technology cannot be ignored. To some people, in some situations, a new way of looking at, and of doing, many things, will emerge. Some will bury their heads in the sand; others will gallop into the fray only to be hoisted upon their own petard. The rest of us, more sensibly, will no doubt try to strike a happy medium.

No, we shall refrain from any speculation about the effect upon the writer's product, although the subject intrigues us. But we cannot so readily ignore the concern that some editors and writers have expressed that they may well have less time to do their *proper* job if a large proportion of their day has to be devoted to the creation of *input* that is automatically fed into *the system*, by means of the *capture* of their *keystrokes*. This concern will cause us to examine the extent to which the author (as for example, reporter), or the editor, is in fact inhibited, or whether he/she may not be "liberated" by the new technology. The burden of accuracy and stylistic consistency may not be so onerous for the future as it has been in the past since there are certain kinds of functions of an editorial nature which computers can perform unobtrusively, and some publishers, or even some editors, may come to appreciate the assistance which can be so rendered.

The absence of a "logical evolution"

ONE FURTHER observation seems appropriate before we plunge into our subject, and that is that the technological changes we shall be discussing did not occur "logically" from a "system's" point of view. These changes started at the 'bottom" and are working their way to the "top." To express it even more colloquially: they began "downstream" and are moving "upstream." Consequently, we shall be obliged to consider the subject matter somewhat in the order in which things came about, rather than to represent these events as if they comprised a rational, total solution. And so we shall begin with our tracing the threads of the story very much (but not entirely) in the sequence in which the events occurred.

1

How Type Composition Differs From Typewriter Composition

TYPEWRITERS produce a form of "cold composition," as distinguished from "hot" type.[1] There are other forms of cold composition which produce character images by photographic or electronic means. But a typewriter produces its images by the impact of a raised letter striking a ribbon and leaving a carbon or ink impression on the paper. Thus a typewriter produces "impact composition."

1. By "hot metal" we mean the process of setting type by casting machines such as the Linotype or Monotype. Both involve the casting of type characters from matrices into which molten lead has been injected.

The term "composition" would be used if the product of the typewriter (or typesetter) were to be reproduced, as in a book or report, rather than to serve as "copy" (or "manuscript") for setting. Composition implies some effort to arrange the material so that it is not only readable, but so that it is presented, within the page frame, as aesthetically as possible.

Character widths [escapement]

The characters on most typewriters are all of the same width. The term "uniform widths" is used to describe the appearance of images so designed that each takes up the same amount of space, whether it is a punctuation mark, a lower-case character, or a capital (upper-case) letter. We also use the term "monospaced" to describe this condition. When a typewriter key is struck, the carriage —including the platen or roller as well as the "paper transport mechanism"— moves from right to left by the width of one character. This is the "escapement" of the carriage, and it is controlled by a gear and pinion device. On IBM Selectric typewriters, it is the typing "element" or "golf ball" which escapes, and the platen remains stationary. In either case it is possible to backspace one keystroke to erase a faulty character, and then to type in another letter with the assurance that it will occupy the same space as the one which was deleted.

However, there are some typewriters (such as the IBM Executive) which produce a typescript consisting of letters and symbols which do not all have the same widths. In this case the escapement mechanism is more complicated. The machine has to "know" what character is called for in order to associate with it its

proper carriage escapement. The range of widths usually varies from a five-"unit" character for capital "M's" and "W's" to a two-unit width for the lower-case "i." If you wished to backspace in order to make a strikeover or replacement, it would be necessary to know the width of the character which was previously typed: each stroke of the backspace key will return the carriage by one unit, and the operator must determine whether to backspace two, three, four or five times.

Proportional type. In the case of "regular" type—that is, hot metal type or type designed to look like hot metal—type characters are *proportional.* This word is used to indicate that not all characters have the same width, but vary in width in order to achieve a better—*i.e.*, more readable or more attractive—appearance, as well as to save space. In general, truly proportional type—which offers a much wider range of alternatives than an "Executive" typewriter—takes up about 40% less space for a given type height than is the case with uniform widths—*i.e.*, monospaced type.

Because of the limitations of typewriter mechanics—especially as related to manufacturing costs—the degree of proportionality, that is, the range of variations in the widths of different characters, is limited, and (with the exception of "strike-on" composition created on the IBM Selectric Composer), typewriter type has a relatively crude and unattractive appearance by typographic standards, even though the IBM Executive variable-width characters may seem more pleasing than some monospaced characters.

This is a sample of Modern Type, the most popular type style for the IBM "Executive" or Electronic Model 50 typewriters.

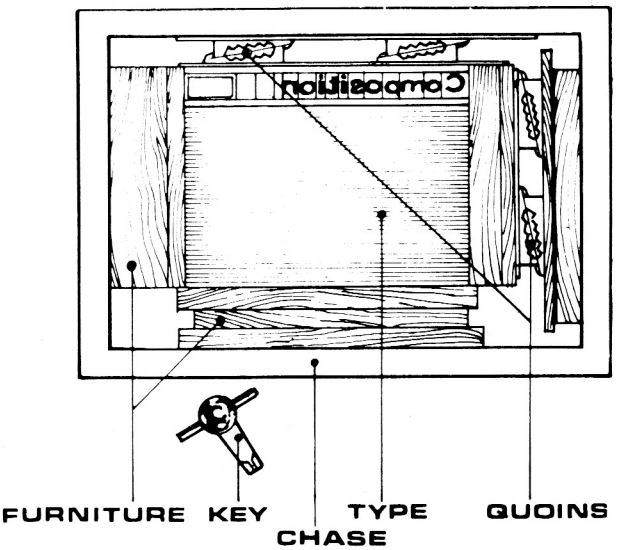
This is a sample of "Courier 10"--which is a monospaced type face commonly used on the IBM Selectric typewriters.

Lines of fixed length

Hot metal composition (and the setting of type from blocks of wood or metal, assembled by hand in the manner that preceded "machine composition") requires that *all lines must be of equal length within a given area of type composition*, such as a page or other "block." This is not to say that they must have the same *apparent* length, but each line must be *filled out* with spaces of non-printing height so that all lines (consisting of characters plus spaces) will be precisely the same length. Individual pieces of type, or slugs from linecasting machines, can thus be assembled into a chase or form, and locked up. These "types" would not "stand on their feet" and would work loose unless all areas within the form were filled solid and squared off, or otherwise subjected to equal tension and pressure from all sides. The type form, which looks like a metal picture frame (*see illustration on the next page*), must be capable of being lifted without the type dropping out.

The need to fill up every line does not exist with impact composition. This need does not exist, either, with other forms of cold composition, where *images* are dealt with rather than physical pieces of type.

Justification. Because lines of type composition in hot metal have to be of equal length, the practice developed of "justifying" type. The tradition emerged centuries ago, long before the advent of machine composition. Since lines had to be of equal length for "lock-up" purposes, the extra space required to fill out the line was inserted *between the words* rather than after the end of the last word or character that would fit on the line. The alternative would have been, of course, to set the margins *ragged* and to add all space needed to even out the line as non-printing space at its end.



At right: Representation of chase or frame showing how type is locked up.

In *ragged setting* the spaces between the words are uniform and the extra space needed to fill out the line is applied at the left, or at the right, or both. This paragraph illustrates text material which is set "ragged right" and we have merely ended the lines at relatively convenient points. To be able to set type attractively in a ragged manner would require more effort than to justify it, because lines should vary in apparent length, but not by too much or too little.

In *justified setting*, the spaces between words are of whatever width is necessary in order to provide right and left margins which are visually aligned. The practice is to have all spaces *within a given line* of equal width (or as close to the same width as possible) so that differences are not discernible. But the space values in one line would only coincidentally be equal to those in other lines of the same text. Occasionally, in order to achieve desired effects or to solve the problem caused by intransigent combinations of words, added space may be inserted not only between words, but also between letters. This is known as *letterspacing*. Here is an example of letterspacing.

How justified setting is achieved. Justified setting therefore implies the use of *variable interword spacing*, and it may or may not also imply the use of letterspacing. However, on a typewriter, justified setting is difficult to produce, and is not particularly attractive. This is due, in part, to the machine's *inability to "put out" fine enough increments of interword spacing.*

You may find it an interesting experiment to try your hand at justifying typewritten text, in the manner shown at the top of the next page.

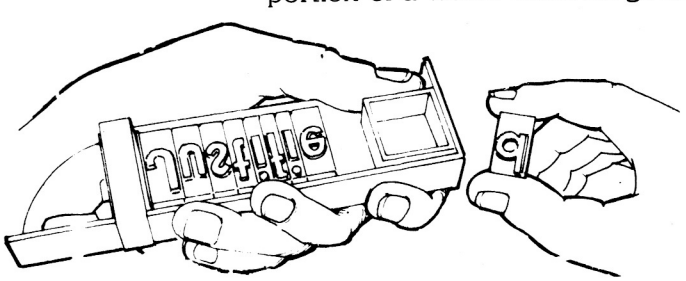
Now I begin to type and watch to see how close I can come to approximating the measure I have chosen. In this case I am placing an "x" at the end of the line to indicate the number of spaces I must add. I will add them at the beginning of the line, from left to right.

But in instances where the typewriter offers some degree of proportionality, it also provides somewhat finer spacing increments, and the result is therefore a trifle more successful.

Now I intend to make these lines justify while setting proportional type on the IBM Electronic Typewriter Model 50. This requires typing the entire block of copy twice. The first time you indicate the amount you are over or under and then the second time you add or subtract units to make the lines fit. It is easier to subtract units than to add them since the smallest increment you can add is a four-unit space from which you can backspace one unit at a time. Now I will retype this material.

How lines are justified in hand-set type

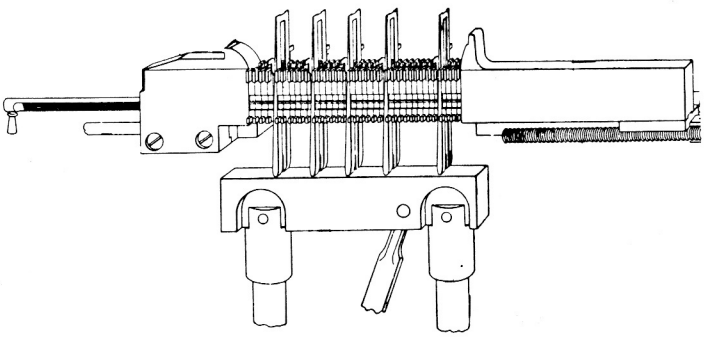
A hand compositor assembles his type characters in a composing stick which is adjusted to a particular line length or "measure." First he fills the line approximately, so that there is no longer any room for another word or logical portion of a word. Then he goes back and adds space or takes out small increments of space between the words, until the line fits tightly. If he "spaces in" he will replace the spaces he has inserted between the words with still smaller spaces, so that the last word in the line can be made to fit. If he "spaces out" he will carry over that last word, and will add more space between the words in the line.



How lines are justified in hot metal [Linecasting]

There are two methods of setting type in hot metal composition. In line- or slug-casting (Linotype or Intertype) justification is achieved in the following manner. The operator keys a space by striking his spacebar. This causes a wedge-shaped spaceband to position itself between the type matrices which end one word and begin another. When the desired number of words in a line has been assembled the spacebands are driven up, mechanically, just enough to

tighten the line, and the line is then cast from these matrices as a solid slug—the individual words being pushed apart as necessary by the spacebands. The spacebar thus inserts spacebands for the purpose of creating *justifying* spaces.



Spacebands are pushed from below to take up the amount of space needed to justify the line.

It is interesting to note that the individual type matrices available on a linecaster, although of varying widths, need not be designed so that there is any fixed relationship between the width of one character and the width of another[1]—except for figures (numbers) which often have to be arrayed in a tabular fashion and are therefore of uniform width. The type designer is thus free to express himself as he chooses in establishing widths of individual characters which appeal to him aesthetically and which offer the most attractive combination of letter forms. However, the absence of such a relationship will cause complications with the act of *keyboarding* is separated from the act of *casting*. This we will discuss in Chapter Six, dealing with Teletypesetting.

1. That is to say that individual character widths are arbitrary and characters are drawn (and molded) to suit the aesthetic judgment of the designer. Nevertheless, it is always possible to express such characters, with some "rounding," as a fraction of a thousandth of an inch.

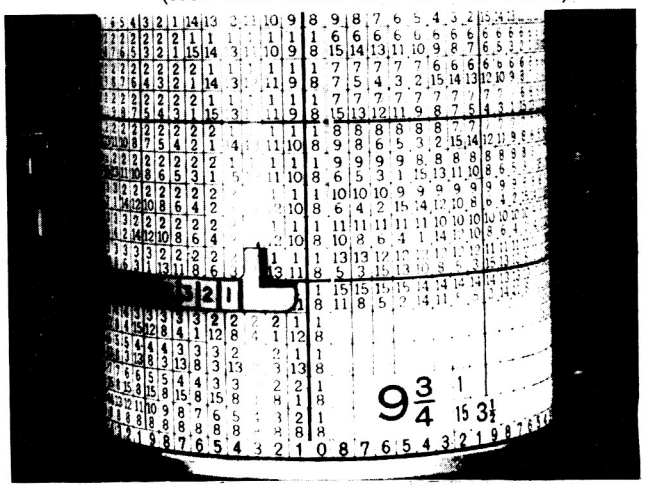
Given, then, the "arbitrary" design of type characters for the linecaster, whatever widths the type matrices prove to be, the spacebands will take up the slack within the line, and the operator can judge how many characters to fit into the line by visual inspection as each matrix falls into the assembler area. If he tries to set too many characters he may remove a few matrices by hand, but his eye can quickly learn to judge when a line is sufficiently full that it will cast properly without excessive interword spaces. The operator must consider not only the apparent width of the array of matrices, but will take into account the number of spacebands, since they are expandable. Obviously, the greater the number of spaces in a given line, the more opportunity there is for the line to be made to justify without extensive manipulation or resetting. Consequently, narrow measures of lines of a given point size are more difficult to set than wide-measure lines, since the former have fewer spaces in them. But for each line, the operator must make an *end-of-line* decision, observing whether the line will or will not justify. If it appears too loose he will add the next word, or a portion thereof, if that word can be divided. His end-of-line decision therefore involves two considerations: *justification* and *hyphenation*.

Monotype composition

The other method of hot metal typesetting is by Monotype. Here the operation is somewhat more complex because it involves two pieces of equipment: a *keyboard* and a *caster*.

The operator at the keyboard cannot see the typeface he is setting. When he strikes a key his action perforates holes in a wide roll of paper tape. With the punching of each key, the units of "set" are summed by the advance of the unit

wheel so indicated. The justification drum starts to revolve 71 units from the end of the line; as each variable space is keyed the justification scale pointer rises one division, and then indicates the justification wedge positions necessary to justify the line. This is explained more fully in Chapter Five, but the point to be made here is that the keyboard must "know" the specific widths of each character, and it is necessary for the type designer to express character widths in conveniently additive values. An 18-unit system was developed so that all type characters could be described with 18 as the denominator and some lesser value (not less than five or six units) for the numerator.



When the operator strikes his spacebar, the mechanism assigns a provisional value to the space between words and tallies the number of such spaces. The justifying potential of the line (unless letterspacing is used) is *a function of the number of justifying spaces within the line.* If only one space appears, it is obvious that the expansion potential of that line cannot be very great. If there are five or six, or more, satisfactory justification is more readily achieved.

By observing his counting mechanism, the keyboard operator can determine when he is within justification range, and he then terminates the line, looking at the justification scale and keying at the end of the line the values required to achieve justification. The paper tape he creates is then fed into the casting machine, reading from the end of the last line and last word forward, thereby conveying the necessary data for the casting of the spaces in that line before the line itself is set. It is also possible to cause the caster to insert tiny amounts of "sidebearing" space between the characters throughout the line. This is a form of letterspacing.

Considering the fact that the Monotype system was developed in the 1880s, it was a surprisingly advanced conception approach.

Both linecasting and monotype composition will be discussed in somewhat greater depth in subsequent chapters.[1] Their only relevance at present is to point out what justification is and how it is achieved, as well as to indicate similarities and differences between typewriter "composition" and hot metal setting. So far, the following observations may be derived from our discussion.

1. See Chapters 4 and 5.

- When physical type characters are assembled (as in the instance of the composing stick of the assembler of the linecasting machine) no counting mechanism is necessary, nor need type be designed according to a unit system. Justification is achieved by adding small increments of space manually or mechanically.

- When keyboarding is separated from casting (or character "imaging"—then in order to achieve justification some kind of counting arrangement is essential. This implies that type character widths must be designed around a common denominator. Also, that a tally of interword spaces must be provided.

- Proportional type is superior, or is thought to be superior, to type with uniform widths. But such type is even more difficult to "count up" that type with uniform character widths.

- To achieve good justification, small increments of space must be available to add between words. The smaller the incremental values, the better the result.

- Impact composition cannot be "justified" without retyping, or without the use of some mechanical or electronic arrangement to capture keystrokes and play them back again with proper spacing added.

There are still other differences between typewriter composition and typesetting which need to be considered, but first let us try to summarize or check out your understanding of what has been covered so far.

Review Questions:

1. What is "cold" composition? What is "hot" type?
2. What is "impact" composition?
3. When is a typescript "composition" and when is it "copy"?
4. What is the relationship between "escapement" and character widths?
5. What do we mean when we say that a machine must "know" the width of type characters?
6. What would you say would be the difference in readability between words made up from characters of uniform width and words made up from characters of proportional width?
7. What kinds of characters take up the most room: monospaced or proportional?
8. What is the physical problem of assembling type characters into lines and locking them up for printing?
9. Is the problem the same for cold composition as for hot? Why or why not?
10. What do we mean by "justification"?
11. How would you define "ragged setting"?
12. What is the relationship between justification and hyphenation?
13. What is a "justifying" space? What is a "spaceband"?
14. How is justification achieved on a linecasting machine?
15. How is justification achieved on a Monotype keyboard and caster?
16. What are the design limitations on Monotype type that do not inherently exist for linecasting matrices?
17. What special problems are posed by the separation of the keyboard from the caster in Monotype?
18. What is "letterspacing"?
19. What relationship does the number of interword spaces in a line have to do with the quality of justification?

2

Further Comparisons Between Typewriter and Typesetter

WE HAVE SEEN that the typewriter produces its "output" copy by impact. Usually its character widths are uniform, but not invariably. Usually its product is ragged setting, and when an effort is made to cause the right-hand margin to look "justified," the interword spacing is awkward because the spacing increments are not small enough. And just as the proportionally-spaced typewriter provides more attractively justified copy than its monospaced counterpart, as you will have observed from the examples on page four, so printers' type offers even greater flexibility in the justification process. Moreover, the letters fit together more compactly so that the eye can more quickly identify words and skim across the page with a minimum of effort. This, of course, is the advantage offered from the use of any form of printers' type—whether by virtue of the hot metal process, or by means of photocomposition.

The standards of good typesetting owe a great debt to the conventions developed by the hand compositor and subsequently imitated through hot metal machine composition. This means, among other things, proportional type and fine increments of spacing, so that the allocation of the extra space at the end of the line can be equalized between words and not seem too gross.

When photocomposition devices came into being they were called upon, for obvious competitive reasons, to offer typesetting solutions which hopefully could achieve the same general aesthetic results as in the case of hot metal machine composition. (This writer believes that good phototype composition is now inherently superior to the best "hot metal" typesetting, for reasons which, we hope, will become apparent as we go along.)

A person with no knowledge of type, but only of the typewriter, is not likely to appreciate some of these fine points until they are made obvious. One of them has to do with the greater readability of proportional type. Another has to do with the manner in which comprehension can be enhanced by the use of *typographic*

emphasis. On a typewriter all you can do is to capitalize or underscore. Perhaps some letterspacing would be possible. You might say: "r e a d t h i s c a r e f u l l y" — and add letterspacing for emphasis to your stock of typographic tricks. But in *typesetting*, and by this we mean *printers'* typesetting, you can use *italics* and **bold face**, too, to lend emphasis. And you can also change type faces and styles. Different type sizes are available to you, and characters may be placed above or below the base line for added clarity. But these tricks cannot be produced by magic, and they are often expensive. Sometimes they require a great deal of hand work. Yet they are features of what we shall call, somewhat snobbishly, *printers' typesetting*—features which lend authenticity, persuasiveness, clarity, cogency, and beauty to the written word. As we shall see, one of the most remarkable features of computerized composition is the tremendous flexibility which it affords in this respect—far more than anything that can be achieved through conventional (that is, hot metal) printers' typesetting.

We shall have occasion, shortly, to consider some of these typographic features. For the moment, however, let us think about some of the other differences between the typewriter and printers' type.

The character set

The "character set" of the typewriter is the repertoire or array of characters that will be found on the keyboard. The typewriter usually provides 44 keys plus a spacebar, a "shift" key, a backspace feature, and some tab stops. Since the shift key implements the upper and lower case modes of the keyboard, it can be said that, for character repertoire purposes, the conventional typewriter offers a total of 88 characters, plus a space. Usually one or two characters (the period and the comma) are the same in both the "shift" and the "unshift" positions. This means that the typewriter most frequently provides a repertoire of only 86 characters, plus the space.

For the English language, most of these characters are universal: there are 26 "alpha" characters, and in their upper and lower case variants this makes up 52 out of the 86 which are available. There are ten numbers, and the uppercase counterparts of these ten numbers are more or less arbitrarily assigned. There is no "standard" keyboard layout in this respect. The author's typewriter has the following:

```
abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
1234567890-=
!@#$%¢&*()_+
½¼;:'",./?
```

How does this character repertoire compare with a printers' type font? First of all, we should remember that some typewriters economize in the use of keys by asking the operator to keyboard a lower case "l" for a "1" (one). On our typewriter there *are* two keys, but the images are still almost identical. Nevertheless,

in printers' type it is essential to be able to differentiate these two, and one would think this should be the case with the typewriter, but it seldom is.

Turning to the "non-alpha" characters: numbers, punctuation marks and the like, perhaps the first need is for what is known as an "em dash"—a horizontal dash longer than a hyphen—which is usually simulated by the typewriter by the use of two successive hyphens. In addition, most typographers would agree that a "nut dash"—a horizontal dash longer than a hyphen but shorter than an em dash—is also desirable. It is used to connect two dates, such as 1888–1960, or to join compound strings, such as "real-time-on-line device." Attention should also be paid to the problem of quotation marks. There are "left quotes" and "right quotes" and "single quotes" and "double quotes." Usually a conversation opens with two left single quotes, which make up one double left quote, and closes with two right single quotes. One right quote by itself is used as an apostrophe, as in the word "doesn't." Single quotes may be used within double quotes: "I heard you say 'ouch' but I thought you were joking." Or, " 'Preventative' is a so-called word I never use."

fiflffffiffl

Some of the other characters, especially some of those above the numerals, vary in their degree of essentiality on a typewriter depending upon the nature of work to be performed. There will never be enough room on a standard typewriter keyboard for all of the characters frequently used, and fractions invariably present a problem. But attention should be called to the fact that in printers' type there are not only many more special single characters, but there may also be combinations of characters as well—such as *ligatures*—where two or more letters are joined into one. We do not have ligatures in the font we are using for setting this book, but you will find examples in the margin.

Why make a comparison between printers' type and the typewriter keyboard with respect to character repertoire? For three reasons: first, because the printer has the problem of converting the typewritten manuscript into type, just as the typist has the problem of describing to the printer what characters are truly intended. Second, because the typewriter keyboard layout, with certain modifications, is used increasingly as an *input device* for typesetting. It is therefore necessary to bridge the intellectual gap between the typewriter and type.

And, as we shall see, there is a third reason to be concerned about the analogy between the typewriter and printers' type: that is the possibility of reading typewriter-produced manuscripts by optical character recognition (OCR) devices, and creating printers' type from the signals so generated by the character recognition devices.[1]

1. *See* our discussion of Optical Character Recognition in Chapter 17.

But in addition to considerations of character repertoire, there are formatting considerations as well, since the typist, the typesetter, and even the OCR device must wrestle with the manner in which the text appears on a page. We have not been concerned, so far, with the problem of *formatting* textual material but it is obvious that there can be a whole host of problems, some of which are created by the very fact of proportional type as opposed to monospaced characters.

Considerations of format

Certain format conditions are so obvious that we do not stop to think about them. Consider, for example, the matter of *spaces*. It is sufficient to have only one kind of space key on the typewriter. (If we ignore, for the moment, the use of tab stops.) We do not need to be concerned about the difference between a fixed space and a variable or justifying space. A paragraph indention is always a fixed space—usually an "em" space. (In this book we are using a two-em paragraph indention.) There is also an en ("nut") space on a linecasting machine. It is equal to one-half of an "em" and usually has the value of the width of a single digit numeral. (Sometimes a "figure space" is specifically used for this purpose, since figures are not always the width of an en.) A thin space is the smallest fixed space increment available on a linecaster. It is used for letterspacing or to set two characters or symbols apart slightly. In photocomposition, many users prefer a fixed space which is capable of redefinition or which may be expressed as the smallest value the typesetting machine can "put out." In fact, on the machine we are presently using, we can call for either a positive (or a negative!) fixed space of virtually any value.

There is a one-em fixed space at the beginning of this particular paragraph. There is a four-unit fixed space between the next two syllables: du Pont. Fixed spaces are necessary to handle indentions required by lists, where you want to line up on a period. The same would be true for positioning certain tables of figures:

1. This is a part of a list.
10. This is a continuation of the list. Notice than an en space was keyed before the "1" in the first entry above. Notice, too, that the "turnover" lines in this paragraph line up in the two entries. We could not have achieved this if we used a variable space after the number and its period, so we also elected to use an en space there. To align the text we had to make allowance, too, for the width of the period. Consequently, the amount of the indention on the turnover lines we had to calculate at 32 units. Vertical alignment is even more complicated when roman numerals or letters are used, as is shown below.

i. This is the first entry to align.
ii. This is the second entry to align.
iii. This is the third entry to align.
iv. This is the fourth entry to align. Setting text in this fashion often requires the use of a tabular program in the "software" of the phototypesetter.

Another format problem is to recognize the *end of a paragraph*. This is what printers call a *quad left* or *flush left* condition. This means, in hot metal, that the rest of the line must be filled out with "quads"—non-printing fixed spaces—in order to make it fit tightly so that it will "lift." And on a linecasting machine an entire slug of the same measure as that of other text in the block must be cast. To do this, the operator will strike his keys a sufficient number of times (interspersed with occasional justifying spaces) to produce the required result—a full, tight line.

Alternatively, a device known as an "automatic quadder" will do this for him. On a Monotype keyboard the operator will also fill out his line with fixed, non-printing spaces. It is not only the end of a paragraph that must be quadded. Single lines, whether flushed left, right or centered, must also be filled with non-printing spaces in hot metal. In photocomp this does not actually have to be done, but the text must nevertheless be correctly positioned in the same manner.

Quadded lines vs. running text

Of course, it is only running text that is "justified"—except for the last line of a paragraph, which is "quadded." But there are also blocks of text consisting of single lines, and these are generally composed with a constant value for the inter-word space—usually about an "en" space between words. Such single lines are generally specified to be set:

Flush left or quad left

Flush center or quad center

Flush right or quad right

Other locations can be achieved by inserting fixed spaces or indentions in the text before the left, right or centered decision is commanded. For example:

Flush left against
a four-pica indent.

Flush right against
a four-pica indent.

Centered within the text defined above.

Note that the fact of quadding the line means that the line does not "run over" (since only that portion which turned over would then be quadded, and the preceding material would be justified).

For example, here is a line which is too large to fit and therefore the first line has been justified, while the second is quadded center.

Here is a runover line where the instruction has been given to quad the second line to the right.

This line is to be set to flush both left and right.
This is sometimes called "quad middle."
This is ..... leadering.
This is also ..... leadering.
This, too, is a kind of _____ leadering.
And so _____ is this.

The function of the "leader"—as opposed to the rule—is to "lead" the eye across the page. But the insertion of a rule (unless an underline) often involves the same formatting problems.

There are literally hundreds of formatting problems which are posed in the composition of even relatively straight-forward text. Suppose a head or caption is desired which will not fit within the measure. Would you take it upon yourself to re-write the head, or would you change the point size, measure or face, or would you try to set the head on more than, say, one line?

Here are some typical illustrations of the problem of setting heads. If you were to use foundry type you would be able to see whether or not the heads fit.

A head set in 36 points:

THIS HEAD WILL N

And so we may center the head:

THIS HEAD WILL NOW FIT

Or reduce its point size:

THIS HEAD WILL NOW FIT

Or find a different combination of size and type face:

THIS HEAD WILL NOW FIT

There is, of course, an infinite variety of formatting options available to the designer who has phototypesetting equipment at his disposal, although some devices offer more flexibility than others. But many format problems present themselves because of the desired intellectual structure of the text. This is usually the case with indentions, of which there are many kinds:

This is known as a "hanging indent"—which means that the first line goes to the margin and all subsequent lines are indented until you reach the end of the paragraph. Then you start over again with the same format, as we shall illustrate here.

Now we have started another paragraph or entry, also in the hanging indent format. This is a common method of presenting lists.

> Then there can be a "left indent" which handles running text by moving it to the right. Often this is done for extract material—something that is being quoted from another source.

Here is an example of a "right" indent." Combinations of indents are, of course, possible—such as a left hanging indention plus a right indent of a certain value. Centered indents are frequently used as well.

The centered indent is perhaps most commonly used, often in conjunction with a change in point size and perhaps, even, type face:

> This is an illustration of material set as if it were extract, centered on the body of the text. In this case we have reduced the point size to ten and we are using a "Megaron" or Helvetica-like type face. We have changed the leading, to twelve points, since the basic face is now slightly smaller. The effect of this presentation is to make the incorporated material stand out.

Another common indent situation is a "run-around," where the text is arranged to flow around some illustrative material, such as a small picture of a syndicated columnist. Such run-arounds are specified as to the amount of indent and the number of lines during which it should continue. In this instance, the run-around will last for six lines. We shall leave a blank space to emphasize the appearance of the format, and then resume setting in the regular width. Another common application of the run-around is in the setting of three-column material, where the illustration desired in the center column will take up more than the width of that column and the extra space required involves the use of a run-around on the right in the left-hand column and a run-around on the left in the right-hand column, with blank space of the same depth in the center column.

The reader may also refer to our "chapter opening" pages—those that begin each chapter—for examples of run-arounds to handle the effect of various "dropped initials." But there could also be ornamentation of some other kind which would require special treatment, especially for chapter opening pages, or to accommodate illustrative material and tables.

Tabular material

Tabular material is always difficult to typeset—much more so than to compose on the typewriter. This is true even though figures have a "monospaced" value. Letters do not, and therefore it is more difficult to align material or even to determine what will fit in a given space. This is especially true if you are performing what is essentially a "blind" operation. You do not see the type until it has been processed through the necessary chemistry, after it leaves the typesetter. On the other hand, the Monotype keyboard operator frequently specialized in tabular work when he could not see what he was setting. (This does not mean that it was easy!) To set tabular material on a linecasting machine is difficult because the spaceband expansion interferes with the alignment of columns. The monospaced typewriter—where you can actually visualize what you are setting—is certainly the simplest way for the novice to proceed. And it will not be an easy task for the typesetter to imitate what the typist has done.

Here is a tabular example. First, as it would be produced on a typewriter, using the "tab stops" to position the columns:

INDUSTRY GROUP	ESTABLISHMENTS	EMPLOYEES
Newspapers	88,094	335,900
Periodicals	2,510	633,700
Book Publishing	1,022	52,000
Book Printing	744	44,700
Misc. Publishing	1,493	31,100

Here is the same table set "in type"—a more formidable undertaking, especially in photocomposition, since the operator cannot see the type until it has been processed through the required chemistry.

Industry Group	Establishments	Employees
Newspapers	88,094	335,900
Periodicals	2,510	633,700
Book Publishing	1,022	52,000
Book Printing	744	44,700
Misc. Publishing	1,493	31,100

Here is an example of even more complex material composed on this typesetter. In such cases formatting ultimately becomes easier than on a typewriter, because once a "format" has been developed it can be recalled by the use of only one or two keystrokes.

Mergenthaler Linotype System 5500

Installation & location	Date first installed	Host hardware	Terminals	Peripherals	Output typesetters	Application
Adelaide *Advertiser* *Adelaide, Australia*		8 Prime 200's, ea. w. 64K memory 6 80Mb disc drives	138 L 400's 6 PageViews	*Mag tape*	2 Linotron 606's	*Newspaper*
Melbourne *Herald* *Melbourne, Australia*	(2/79)	19 Prime 200's, ea. w. 64K memory 16 80Mb discs	9 PageViews 200+ L 400's		3 Linotron 606's	*Newspaper*

And so we become aware that there are some peculiarities to typesetting which arise out of the handling of spacing—especially so-called "area composition." Some of the problems which occur result from the proportionality of type, and some of them arise because of the use of "justification" because we can no longer assume that interword spaces all have equal value. In addition, there are, as we have just seen, special challenges in "formatting" text with indentations, tabular material, run-arounds, and even more complex situations which we shall encounter in the "making up" of pages.

Identification of problems and situations

The most difficult task for a person entering the field of programmed typesetting is to be able to define the problems to be faced and the solutions desired. There are problems of *system*, *equipment*, and *typographic requirements*. We have endeavored, at the outset, to suggest some of the differences between typewriter manuscript preparation and typesetting in order to create an awareness of some of the unique aspects of the challenge. We shall not dwell at any length upon the way in which composition is achieved by conventional hot-metal methods, except to provide some background for understanding photocomposition. We are more interested in making use of modern technology to accomplish similar results more easily, more economically, and more precisely—effects which otherwise might demand a high level of craftsmanship. Nevertheless, technology is no substitute for art, since it can be used to produce bad results as well as good—and it may take something of a connoisseur to know the difference.

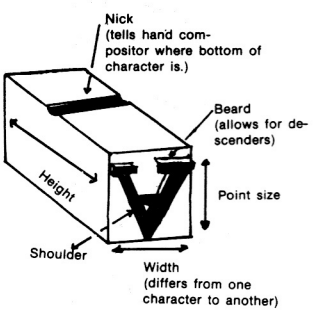
Review Questions:

1. What do we mean by "character set?" What is the character repertoire of your typewriter? What "characters" does it not have that you need frequently?
2. We have used the term "alpha" and "non-alpha" characters. What are they?
3. What is a ligature? Do you find any in the text of this book?
4. We refer briefly to OCR. What is this?
5. Describe the different types of fixed spaces used in typesetting.
6. What is the difference between a fixed space and a justifying space?
7. How do fixed spaces help you with format problems—especially lists and tables?
8. What problems do you encounter in centering a caption?
9. What problems do you encounter in "leadering"?
10. What are the forms of indention which are most commonly used and what is their function?
11. Why is the setting of tabular matter more difficult when type is proportional than when it is monospaced?
12. What are some of the elements of composition?

3

Characteristics of Type— Basic Concepts

ALTHOUGH NOWADAYS much typesetting is done by means of photocomposition devices, the design of type characters owes so much to their hot metal heritage that we have to begin with an explanation of certain features of "foundry" type—that is, individual pieces of type as they were originally cast by hand.

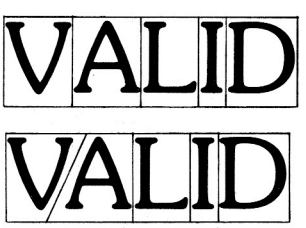


Rectangularity

The first important fact about hot metal type is that each character has to be contained within a block of metal with four square corners, so that each piece can be put together with each other piece—both with respect to characters *within* the line, and also taking into consideration *adjoining* lines. If type were not so designed, it could never be "locked up."

Character widths [escapement]

If each type character must be contained within a rectangle, it is not possible to achieve certain aesthetic effects which provide a closer "fit" with respect to certain character combinations. Consider, for example, the relationship between a capital "V" and a capital "A."



Mortising

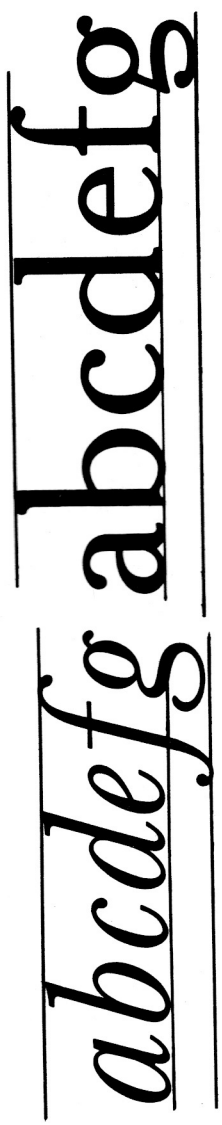
In the margin you will see what these two characters look like side by side, if they "sit" within their own rectangles. In hot metal, if you want a more intimate fit between two characters, you have only two courses to follow. One is to "mortise" the type—that is, to saw off portions of the type body, as illustrated, to bring the letters closer together. The other is to design character combinations as one piece. These become "ligatures," "digraphs," or "logotypes."[1] Ligatures consist of two or more characters created as one, as is the case with the *fl* and the *ffl*, or the *ct* in some French type faces. "Logos" include characters which may be fitted close together, or may even overlap, or they may consist of several letters which

1. Digraphs consist of combinations of letters or letters and punctuation where one matrix is used to contain both images, designed as one entity.

are separate and distinct but in a unique "corporate image" type face. Yet they may occupy only one piece of type or engraving.



In Monotype, moreover, it is sometimes possible to cast characters so that—in certain "display" sizes—the face of the type takes up more room than the body, thus permitting some overlapping. This is illustrated by the "Q" in juxtaposition to the "R" as shown in the left-hand margin. Often, in hot metal, special "kerning" effects can be achieved by treating certain character combinations as if they were ligatures, by placing two or more characters on the same body and casting them from one matrix. On the linecaster this always requires the hand insertion of a "pi" mat, which must be placed in the assembler, and then removed from the distributor. One such special combination, which you will find in such magazines as the American edition of the *Readers Digest*—set in hot metal until 1977—is the use of "To"—where, as you can see, the "o" has been tucked under the letter "T."



Other similar digraphs could include Av, Aw, Ay, Ta, Te, Tu, Va, Ve, Vo, Vu, Wa, We, Wo, Wu, and Wy, as well as Ya, Ye, Yo, and Yu. In italic, other combinations suggest themselves, such as *Fa, Fe, Fo, Ha,* and even *Pa* and *Pe*. Very few typesetters exhibit this much concern about the fitting of lower-case or upper- and lower-case characters in text material.

Point size

In general, the *height of the rectangle* containing the individual characters *determines its point size.* In other words, when we refer to the size of type, such as ten-point or twelve-point, we are really not referring to the height or size of the character itself, but rather to the height of the piece of metal which is *common to all characters within the font.* Individual character *widths* will differ, but the *height* of the metal block will be the same in every case, even for the periods and commas.

The height of the character *itself*, however, will vary from one type design to another, *even for designs in the same point size*. This is because type is designed to meet certain objectives, according to the purposes for which it is to be used. For some purposes, for example, it is desirable to have lower-case characters with a large *x* or *z* height. This means that the ascenders and descenders are shorter in proportion to the body (or *x*-height) of the type. For other purposes, longer ascenders and descenders are desired, and of necessity the *x* or body height of lower-case characters would have to be smaller. You will notice, from the Monotype 72-point samples in the left margin, that the *x* heights do indeed vary from one type face to the next, and also, when you mix two different type designs (in hot metal) of the same point size on the same line, they will usually not "base align" since each is drawn to accommodate its own descenders. In other words, each face "sits" on its descenders (with allowance for a projection on type mold or stem to protect the image from breaking off). And because "cap" heights are determined by the location of the base line, capital letters in some type designs are taller (for a given point size) than others. (Again, see illustrations in the margin.)

Our examples show, therefore, that different type faces will take up the same white space in a vertical direction, but will have different base lines and different *x* or *z* heights. And, of course, character widths will also differ.

It can be readily seen, then, that "point size" is, in a sense, an abstract concept which relates to the amount of vertical white space necessary to accommodate the distance from the lowest descender to the highest ascender. It does not describe the height of either an upper- or lower-case character and, in fact, capital letters usually average in height somewhere between 60 and 66% of point size. (This is an important fact to remember!)

Leading

Before we define "points," "picas," and other printers' measurements, we will have to muddy the waters still further. Suppose you want to determine the point size of the text of a given sample of typeset matter. Can you measure the distance from the top of one ascender to the bottom of a descender of an adjoining character and find out what the point size of the type is? Probably not. The scale is usually too small, and most often you can't find two such characters close enough together to get a good measurement.

What, then, if you attempt to measure the distance from the base line of one line of type to the base line of the succeeding line? Here, again, distances are usually too small to permit a definitive measurement. If you take a pica gauge and lay it along this text, for example, so that the beginning marker is at the base of the first line of this paragraph, and then count down ten lines, you should get a reading of 130 points, or ten picas and ten points. This is because we are setting this text on 13-point "leading," so that 10 lines should equal 130 points. (We are, in fact, setting type which is eleven points in size.) If the leading were twelve points, then ten lines should give you a reading of 120 points, or exactly 10 picas. You must allow, of course, for some minimal shrinkage or expansion in the platemaking and printing processes.

But notice that we are talking about the *leading* and not the point size. Nowadays, the term "leading"—pronounced *ledding*—means the distance between base lines of running text. The term is derived from the practice of inserting strips of lead between lines of type, when composed by hand, to space them out so that they would be farther apart. Your eye will have trouble picking up the next line on fairly wide measure material if it is set "solid." *Solid setting* means setting with no leading or extra white space between the lines. Hence the practice of separating the lines by inserting strips of lead, or "leading," between the lines when you do not want to set solid. Later it became possible, with mechanical hot-metal typesetting machines, to cast, say, a ten-point character on a twelve point body or "slug," thus building the "lead" into the type height—but not the width—of the character itself. Obviously, therefore, *measuring the distance between lines will not tell you what the point size of the type is.* It will only tell you what the leading is. The fact is that only an expert can look at type—even measure it—and be absolutely certain what the point size is. This is especially true today, since,

as we shall discover, the appearance and dimensionality of photocomposed type does not correspond precisely with the appearance of hot metal type. Moreover, one designer's hot metal face—*e.g.*, Times Roman—will differ from that created by another supplier.

While we are on the subject of leading, let us provide you with several examples, showing how this particular type looks when the same point size is set to different leadings. We are using an 11-point type face for the body copy in this book. Now we will show you what the running text would look like if the lines were set closer together or farther apart.

Here we are setting 11-point type on a "body lead" of ten points. Of course this would not normally be possible in hot metal, but it can be done in photocomposition. (Sometimes by mistake!)

Now we are setting the same 11-point type on a body lead of ten and one-half points. You will find this a little more readable, but the ascenders and descenders will still overlap somewhat. To illustrate this, let us set one line with nothing but characters with descenders followed by a line with nothing but characters with ascenders, so that you can judge for yourself if the effect would be acceptable.
qqqqqq gggggg yyyyyy qqqqqq gggggg yyyyyy pppppp qqqqqq pppppp gggggggg
dddddd bbbbbb hhhhhh kkkkkk llllll dddddd bbbbbb hhhhhh kkkkkk llllll dddddd

Now we are setting 11-point solid. Even so, there can be so little space between the descenders in one line and the ascenders in another that they almost overlap. You will probably agree that lines of this width should be set farther apart for readability. And it would certainly be difficult to make corrections in text by removing the offending lines and cutting in new ones, with a razor blade.
qqqqqq gggggg yyyyyy qqqqqq gggggg yyyyyy p ppppp qqqqqq pppppp gggggggg
dddddd bbbbbb hhhhhh kkkkkk llllll dddddd bbbbbb hhhhhh kkkkkk llllll dddddd

Now we are setting eleven point on 11½ lead, which does separate the text a bit more, but it is still pretty close together. Text is seldom set "solid" and for larger sizes, in general the larger the size the more leading is appreciated.

Now we are setting eleven on twelve and you will see how much better this looks. But we suspect that you would not want to read an entire book set with only one point of leading. Two points of leading are more or less customary for book text setting.

As a matter of fact, we decided that since we wanted to make the text quite readable, and therefore felt that eleven-point type would be preferable to ten, we would use the conventional two points of text body leading, and hence chose to set "11 on 13." There is no purpose in making the reading of this book any more of a labor than it has to be. Do you agree that we were wise in deciding to set the text 11 on 13? Your reactions on this point as well as on the choice of type face and general appearance of this book would be welcome. But we ask you to bear in mind the purpose for which the book was written. Don't forget that economic considerations—how many pages the book would make if set less or more densely—also need to be taken into account. The more pages, the more expensive. Also, the heavier!

Picas and points

Now we must return to the problem of type measurement. We have learned that the size of type cannot readily be determined by examining the characters themselves, and certainly cannot be ascertained definitely by measuring the distance between lines of text unless we have already been told (or can figure out by the way the ascenders touch the descenders) that the text is set *solid*.

While we may have trouble identifying a given point size, there are obvious differences from one size to another. The sizes range from very tiny characters, perhaps three or four points "high," to very large sizes—perhaps 72 points, or as much as 96 or 128 points in the case of large banner headlines in newspapers.

Text type tends to range from nine points, in the case of most newspapers, to twelve or thirteen points, in the case of books. Ten or eleven point type is usually very readable, especially when "leaded." Large-print books, for the near-sighted are usually set in 18 point type.

We shall now show you some samples of different type sizes. Please bear in mind that they are set by photocomposition, and from one character master image only. Later you will understand that this means that they will not necessarily correspond with hot metal faces. Type faces for hot metal are usually redesigned for each point size or, more likely, for each range of point sizes. This is more infrequently the case for photocomposition. When there is different art work for different point sizes, ascenders and descenders tend to become more pronounced, and the *x* height becomes proportionally smaller as the type increases in size. Moreover, the "side bearings," or white spaces between (non-letterspaced) characters become smaller and smaller in relation to the individual character.

Here is an example of six point type in the basic design of our text face. We are setting it six point "solid." We think this type looks somewhat smaller than "conventional" hot metal six point type, which means that because it was designed to look its best at around ten points, the *x* height is probably a little too small at six, and the ascenders and descenders too large, proportionately. The type also fits a little more loosely than at larger sizes.

As we move to seven points the type probably more closely resembles a "true" hot metal design, but, still, not quite.

In eight point, the type grows even more like its hot metal counterpart, or so it seems to us. Obviously a great deal of individual judgment enters into one's evaluation of type, which is highly personal, at best.

An interesting fact about this particular typesetter is that it offers an 8½ point size, which we are still setting solid—that is, on 8½ point leading.

At nine points the face looks like this. You can also see that fewer characters fit on a line. As we said, nine points is a common size for use in newspapers, but it is quite often leaded to 9½.

Now we are setting this sample in 10 point type, which is the size most commonly used for book composition, except that the measure would be wider, and the face would be set with some leading.

But we shall show you 11 point as well, since the book is set in this size.

Here, now, is the 12 point example, and again less type fits into the line. It is hard to justify 12 point type in a 13 pica line length.

Thirteen picas is certainly too narrow for 13-point type. Now, so that you can examine the same type in larger point sizes, we will show you 36, 48 and 72 point type in the right-hand margin.

Abc

Abc

Abc

Getting back to the subject of picas and points, now that you have seen how type looks in various point sizes, please bear in mind that not only do we measure type in points, but we express the line length, and even the page depth, in picas and points. (It has recently been suggested that we really should do so in millimeters, defining the unit, called *d* as 1/10 of a millimeter, to specify type size, body size, line spacing, and width of margins.)[1]

1. See paper by S. J. Heden of Stockholm in the December, 1969 INCA-FIEJ R. I. Bulletin. This organization is a federation of European newspapers.

There are twelve points to the pica. When we talk about 120 points we are also talking in terms of ten picas. If we say 10½ picas we also mean 126 points. In some computer programs one may designate line lengths and vertical dimensions in decimal terms. Obviously 10.5 picas is also 126 points. Recently computer programs have given the user facility to express his measurements in inches or millimeters, and the program itself makes an internal conversion into points. Conversely, there are programs which, unbeknownst to the user, work internally in thousandths or ten-thousandths of an inch.

A "point" in the Monotype system is 0.0138 inches. A "point" in the Linotype (hot metal) system depends upon the adjustment of the knife blade. Some writers have asserted that a Linotype point is most frequently equal to 0.014 inches. While 72 points in the Monotype system would equal .996 inches, 72 points in the Linotype system might be equivalent to 1.008 inches. Nowadays, with photocomposition devices, most machines available in the market are designed to give you 72 points to the inch, more or less precisely, so that the correct definition of a point for photocomposition now tends to be 0.013888...

However, on all photocomposition devices, there is a "rounding" problem, and "escapement"—in the last analysis—is handled in absolute rather than in relative terms since character laydown is mechanical for second-generation devices. Even for electronic devices it must be expressed in absolute terms for movements of the beam which sprays electrons against the face of the CRT character-generating tube. There can therefore be very minor variations in the width of a line, as you change from one point size to another, but these variations should be infinitesimal, and sometimes may be compensated for in the electronics of the device itself. It is unlikely that there can ever be any uniformity in the incrementation of these mechanical or electronic movements, but standardization of a common denominator or unit system is devoutly to be wished.

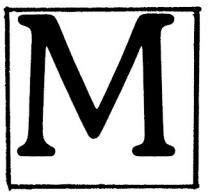
Picas.—As already mentioned, a pica equals twelve points, and the pica is the typesetting measure in the English speaking world. In Germany and France they use a "Didot" (pronounced *dee-do*) system. The Didot point is larger than the English point, so that about eleven of them (instead of twelve) equal a pica. Twelve Didot points equals one "Cicero."

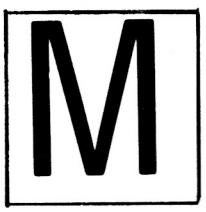
Ems and ens

Now that we have dealt with picas and points, which are *absolute* measurements, we must turn to something which is somewhat more elusive. This is the

printers' "em." (In Britain, a typesetter may say "em" when he means "pica." He is assuming, of course, unless he says otherwise, that he is talking about a "pica em.") But as we use the term more generally, an "em" *can be any point size at all.* It is a "relative" unit of measurement—relative to the point size, that is. Let us try to make this concept totally understandable, since it is vital for the comprehension of typesetting and its technology.

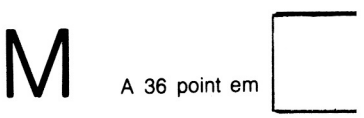
Students of printing (that is, composition—since pressmen don't usually know or care what ems, picas and points are) are taught that an "em" is the space that a capital "M" would take up in hot metal—both as to height and width. The illustration in the right margin should show that since the letter "M" has no descender, the assumption must be made that a proper allowance has been given for the descenders that would be associated with other characters in the font. Hence an "em"—so far as height is concerned—would not be the height of a capital "M" but could be equal its width, making due allowance for the necessary hot metal "side bearings." One common definition, therefore, is that an "em" is the square of the point size, whether or not the type designer always in fact makes the letter "M" as wide as the point size.

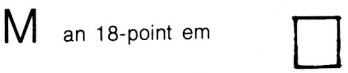




It is frequently the case that the letter "M" is as wide as an em space, but this is not invariably true. Certainly there are "extended" or "condensed" faces where it is not true, as can be seen from the marginal illustration.

But we do not need to be concerned about whether or not the letter "M" is or is not an "em" wide. The "em" is principally relevant as a means of describing type *width.* An em space is as wide as the true point size you are setting (excluding extra leading, of course). If you are setting ten point type, an em equals ten points. If you are setting 18-point type, an em equals 18 points. But if you want to think of em as an area rather than a width, the marginal diagram will help clarify the relationship between the respective sizes of ems. Note that one 18-point em takes up one-fourth of the *area* of a 36-point em.





There is an awkward problem, however, that we must at least acknowledge at this point. That is, that the "set" of type is not necessarily the same as its point size. It is possible to set ten point type as if it were nine points wide. In such a case, considering the em as relevant primarily as a measurement of type width, the ten point type face would have a nine-point em!

An "en" space—sometimes called a "nut" while an "em" is called a "mutt" (because the words em and en sound so much alike and instructions are often conveyed verbally)—is one-half the width of an em. Hence, in ten point type (ten set) an em is ten points wide and an en is five points wide. *Figures*—namely 0,1,2,3,4,5, etc.—are usually, but not invariably, the width of a "nut" or "en" space. For many type faces, lower case characters tend to *average* the width of an en, although some will be larger and some will be smaller. It is often customary to express the width of a certain type face in terms of its lower-case character count, which usually means the number of lower-case characters which will take

up an em, and the number might be 1.9, 2, or perhaps 2.1. For the face in which we are now setting, you may be interested to make a comparison:

01234567890123456789012345=26 ens.
abcdefghijklmnopqrstuvwxyz=the 26 letters of the alphabet—(slightly less than 26 ens.)

Em and en spaces are "fixed" spaces, even when they are "relative." They take on a different width when you move from one point size to another, but they are fixed in the sense that for a given point size they retain the value assigned to them. They are therefore not to be confused with "variable" spaces or "spacebands" which are used to achieve justification. We shall be discussing spacebands (or variable interword spaces) in the next chapter.

With a conventional typewriter you can begin a paragraph with an indention of one or more spaces, without stopping to think whether you are using fixed or variable spaces (since there are none of the latter). In typesetting you must remember that there is a difference.

Another fixed space is called the "thin" space—again deriving its name from hot metal, specifically from the linecaster. A "thin" space is equal in value to one-third, one-fourth, or one-fifth of an em, depending upon the specific application. Nowadays typesetter manufacturers have often found it more convenient to assign a value to a thin space which makes it equal to one-half of an en, or to be user-definable. This may not be possible in some systems. It would often be convenient for the thin space to be the width of a period, to help in the alignment of tabular material requiring the use of decimal points.

What is the significance of the em concept?

Of course, em spaces are relevant to describe fixed spaces, but they are much more relevant in that *they are used as the common denominator for the design of type faces*—and for the consequent calculation of justifying values. This brings us back to the concept of "escapement."

In the first chapter we indicated that printers' type is proportional, so that characters may vary in their widths. But it also becomes apparent that it is necessary in many cases to express these widths in terms of some common denominator. The Monotype system, developed in the 1880s and 90s, required the use of such a common denominator, as we have mentioned several times. This is because the keyboard is separated from the caster. The keyboard operator needs some method of counting so that he can tell when he has composed a line. Tolbert Lanston, inventor of the Monotype, could have devised a counting system which expressed widths of characters in terms of thousandths of an inch—especially since his device, being mechanical, ultimately had to operate in those terms—but apparently he felt that such a system would have been more difficult for ready comprehension by the keyboard operator. Moreover, it was certainly

useful to have a counting system which would work in the same values regardless of the point size you were setting in. So, Monotype fonts were designed in such fashion that every character width is expressed as a certain number of eighteenths of an em. An em space is, of course, equal to 18 units. A nut space equals nine units.

The width of a unit will of course depend upon the point size of the type. If you were setting 18-point type, an em would be 18 points wide and 1/18th of an em would then equal one point. Hence, in nine-point type, 1/18th of an em will equal one-half point. In six-point type an em will have a value of six points and an en will equal three points. One-eighteenth of an em will equal one-third point.

It is possible to express the value of a line width or measure in unit increments of a given point size. If this measure is 32 picas, or 384 points, there are 34.9 eleven-point ems in the line, or 628.2 relative units on the basis of 18 units to the em. (Our typesetter machine counter rounds this off to 629.)

Other relative unit systems

Today, for many phototypesetters, the 18-unit system has been supplanted by something which provides an even greater flexibility for the type designer. Commonly, type is designed, and widths are expressed, in a "half-unit" system, which means, in effect, a 36-to-the-em counting arrangement. Another common system provides 54 units to the em, thus assigning three units where only one served in the 18-unit counting convention. There is also a 64-unit system, and even some based on 1/100th or 1/200th of an em. One supplier has constructed an "absolute" system which expresses width values as 1/64th of a pica—each unit being called a "piclet"—with subsequent, smaller units designated as "triclets." The finer the unit system—that is, the larger the denominator of the em fraction—the easier it is to adapt non-unitized hot metal fonts to a unit system. It is at least partly for this reason that Mergenthaler moved from the 18-unit system to one of 54 units for the V-I-P typesetter, to simplify the process of converting its hot metal library of type faces to the V-I-P without the need to redraw those characters whose general shapes and side bearings would otherwise appear ungainly when reconciled to the 18-unit common denominator.

Today the unit system serves a variety of purposes:

- It provides a method of reckoning or "count-down" to assist the operator in the making of end-of-line decisions when inputting by means of a counting keyboard.
- It informs the typesetting machine how far to "escape" so that each character will take up the appropriate amount of space. If the typesetting machine had no way of "knowing" the width of the character, it would either have to set all letters on top of each other, or else it would have to escape an arbitrary amount, like a monospaced typewriter—and this would, of course, not do justice to the design of the type.

- Relative unit systems are used by type designers as benchmarks for the proportioning of characters in new faces.

- Storing the information about relative widths makes it possible to describe character widths conveniently in table form, without having to deal with the very high number of thousandths of an inch (or fractions of an inch) that might be needed for a system that dealt with absolute values.

- Finally, unit systems provide a method of equating changes in measures and point sizes, so that more complex matter can be composed. True, an absolute system can also achieve this purpose, but we must, somehow, be able to translate from one point size to another with minimal calculations.

Photocomposition sizing alternatives

As we shall see, because of photocomposition, a wide range of point sizes is generally available to the user. Sizing is achieved by the use of lenses which enlarge or reduce characters from their original or master image sizes.[1] In some cases, several different master designs will be used, with the intention that the type will only be set within certain size ranges. In other cases, however, one master, perhaps designed to look best for a twelve point face, will be reduced to as little as six or eight points, and increased to as much as 48 or even 72 points. Naturally, under such circumstances, type so enlarged or reduced will not "look like" hot metal type designs in all sizes. Generally, this will mean, too, that the resulting composition of some material is not as attractive or artistic. But this is not always the case. Just because the type images may look different does not necessarily imply that they cannot be effective and satisfying. The general appearance will depend upon many things, some of which we shall explore in subsequent chapters of this book. But the reader should bear in mind that photographic enlargement or reduction further complicates the problem of identifying point sizes of type and of specifying desired sizes to achieve certain effects. Often, for example, it will require the use of an 11-point photographic face to arrive at the same general appearance—that is, so far as the *x* height is concerned—as a 10-point hot metal face. But this is not invariably true.

1. We mean the size at which the image is stored within the photo unit—not necessarily the size of the image as originally conceived. In fact, an image master may well have been designed to look its best at ten points, and yet be stored, since it would take up less space, as a five point character on a type disk. When enlarged back to ten points with a 2x lens, it would then take on its original characteristics.

Another peculiarity of photocomposition is that, generally speaking, all sizes of type are designed so that they are *base aligning*. We have seen that this is not so with hot metal. If you set a nine-point sans serif lead-in head in the same line with a nine-point Caledonia roman, you may find that the base lines do not correspond. But with photocomp, the machines are built so that they will *will* align or focus the character on an intended optical base line. Thus it is even possible to combine different sizes of type within the same line and preserve base alignment. Shown immediately below is perhaps an extreme example of such a mixture of sizes and faces:

This is illustrated here for all *to* see.

There is, however, one exception. There are certain kinds of phototypesetting devices that enlarge their photographic master images by the use of a "zoom" lens system, rather than by shifting from one fixed-focus lens to another. Where fixed-focus lenses are used they can be adjusted so that the base alignment of different sizes can be achieved. But when "zoom" lenses are used it is not always possible to accomplish this. What you may get is illustrated in the right margin. Here we show base alignment and the effect of non-base alignment due to magnification.



However, there is an increasing tendency to avoid this problem, even with zoom lens systems—in machines of recent design, such as the one we are using to set this book. The problem can be avoided if only a quarter of the lens is used. If this is not done the effect you will get is what you would see if you focused an image through a 35 mm projector onto a screen and then backed up the projector in order to get a larger picture. You would have to adjust the "base line" of the picture to fit the screen, since the reference point for such lenses remains in the center of the lens.

Type styles and faces

It is not our intention to turn this book into a treatise on type faces and type design. However, we *do* need to presume some interest in typography on the part of our readers. Therefore we shall use such terms as *serif* or *sans-serif* without further definition. We shall perhaps refer to *Baskerville*, *Times Roman*, *Helvetica* and *Univers* if we find the need to do so. The reader should take the trouble to learn something about type faces elsewhere, since our interest in this book is primarily with technology.

However, a few very superficial observations about type faces and type design may be in order. We often use the term "roman" to refer to any face used for the presentation of text matter. This is to distinguish "roman" from italic, or bold, or bold italic. In this sense, if we were setting text in a sans serif type face, we might regard a command to "return to roman" to mean to go back to the normal version of that sans serif face, as opposed, say, to the italic version. Actually, however, "roman" is a particular kind of type design copied three centuries ago from contemporary humanistic manuscripts. Consequently, the type was conceived to imitate the strokes of a broad-nibbed pen held at right angles to the paper. Then, "it is clear that a vertical stroke will be of the same thickness as the pen's edge, and that a horizontal stroke will be a thin line, while curved strokes will vary in thickness from the one extreme to the other. In an o the two thickest parts will be directly opposite each other—due east and west, so to speak. In such a letter we describe the distribution of the thicks and thins, or the distribution of weight as vertical; we speak of the vertical colouring, stress, or shading. Holding the pen at right angles to the paper is a strained attitude, and we find that an angle of 45 degrees more or less was more usual. This gives a diagonal shading; for example, the thickest parts of the o are diagonally opposite—north-east and south-west, or in some cases, even north-east-north by south-west-south. Such shading is characteristic of the early romans and of most of the book

types in use today. . . . All book types show some variation of thick and thin; in the fifteenth century it was slight and gradually became more pronounced, until it reached the extreme in the nineteenth century when they became called hair lines."[1]

1. *The Encyclopaedia of Type Faces*, by W. Pincus Jaspert, W. Turner Berry, and A. F. Johnson, Barnes & Noble, 1970. Pages not numbered, under "Classification."

Actually, italic is part of a complete roman font. "Such is the sad fate of this once beautiful letter, originally an independent rival to roman. In origin it is a cursive form of the Italian hand, on which roman types were based, and should properly be called *Cursive*, as in fact it is called in Germany. . ."[2] Jaspert, Berry and Johnson, in their classic compilation of type faces, extracted from *British Standards 2961:1967* this categorization of type faces:

2. Same citation.

Classification of Type Faces

1. Humanist

(Such as Verona and Centaur)

Type faces in which the cross stroke of the lower case e is oblique; the axis of the curves is inclined to the left; there is great contrast between thin and thick strokes; the serifs are bracketed; the serifs of the ascenders in the lower case are oblique.

When jobs have type sizes fixed
When jobs have type sizes fixed quickly
ABCDEFGHJKLMNOPQRST

Centaur

2. Garalde

(Such as Bembo, Caslon, Garamonde)

Type faces in which the axis of the curves is inclined to the left; there is generally a greater contrast in the relative thickness of the strokes than in Humanist designs; the serifs are bracketed; the bar of the lower case e is horizontal; the serifs of the ascenders in the lower case are oblique.

Bembo

abcdefghijklmnopqrstuvwxyz £1234567890
ABCDEFGHIJKLMNOPQRSTUVWXYZ&.,:;'' !?-()[]
ABCDEFGHIJKLMNOPQRSTUVWXYZ

3. Transitional

(Such as Baskerville, Bell, Caledonia)

Type faces in which the axis of the curves is vertical or inclined slightly to the left; the serifs are bracketed, and those of the ascenders in the lower case are oblique.

abcdefghijklmnopqrstuvwxyz£1234567890
ABCDEFGHIJKLMNOPQRSTUVWXYZ&.,:;'' !?-()[]
ABCDEFGHIJKLMNOPQRSTUVWXYZ

Baskerville

4. Didone

(Such as Bodoni and Corvinus)

Type faces having an abrupt contrast between thin and thick strokes; the axis of the curves is vertical; the serifs of the ascenders of the lower case are horizontal; there are often no brackets to the serifs.

the margins of error widen unless there
When jobs have type sizes fixed quickly
ABCDEFGHJKLMNPQRSTUVWXYZ

Bodoni

5. Slab serif

(Such as Clarendon, Rockwell)

Type faces with heavy, square-ended serifs, with or without brackets.

abcdefghijklmnopqrstuvwxyz.,:;'' ! ?-()[]
ABCDEFGHIJKLMNOPQRSTUVWXYZ&
£1234567890

Rockwell

Classification of Type Faces
(Continued)

6. Lineale

Faces without serifs, often called "Sans-Serif." Lineale type faces with 19th century origins. There is some contrast in thickness of strokes. They have square curves, and curling, close-set jaws. The R usually has a curled leg, and the G is spurred. The ends of the curved strokes are usually horizontal.

a. Grotesque
(Such as Grotesque, Sans No. 7, Headline)

quickly margins of error widen
ABCDEGHJKLMNOPQRSTU

Lineale type faces derived from the grotesque. They have less stroke contrast and are more regular in design. The jaws are more open than in the true grotesque and the g is often open-tailed. The ends of the curved strokes are usually oblique.

b. Neo-grotesque
(Such as Univers and Helvetica)

Grotesque

abcdefghijklmnopqrstuvwxyz£123
ABCDEFGHIJKLMNOPQRSTUV
Univers

Lineale type faces constructed on simple geometric shapes, circle or rectangle. Usually monoline and often with single storey a.

c. Geometric
(Such as Futura, Eurostyle)

ABCDEFGHIJKLMNOPQRSTUVW
abcdefghijklmnopqrstuvwxyz Futura

Lineale type faces based on the proportions of inscriptional Roman capitals and Humanist or Garalde lower case, rather than on early grotesques. They have some stroke contrast, with two-storey a and g.

d. Humanist
(Such as Optima and Gill Sans)

When jobs have type sizes fixed
quickly margins of error widen
ABCDEFGHJKLMNOPQRSTUV Gill Sans

7. Glyphic

Type forms which are chiselled rather than calligraphic in form.

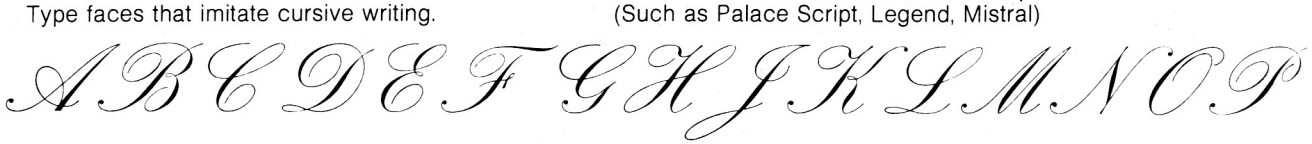
(Such as Latin and Albertus)

ABCDEFGHIJKLMNOPQRSTUVWXYZ&ÆŒ
abcdefghijklmnopqrstuvwxyzfiflffffifflæœ
Albertus

8. Script

Type faces that imitate cursive writing.

(Such as Palace Script, Legend, Mistral)



Palace Script

9. Graphic

Type faces whose characters suggest that they have been drawn rather than written.

(Such as Libra, Cartoon, Old English)

ABCDEFGHIJKLMNOPQRSTUVWXYZ&ÆŒ
abcdefghijklmnopqrstuvwxyzæœfiflffffifflchckllſſiſſſtʒß
Old English

What is a font?

We have made occasional reference to a "font" of type. The term "font" means different things to different people. It may mean the characters available on the linecaster, and here the number of characters is limited by the number of channels on the machine. It may mean the number of characters on a grid or a section of a disk for a phototypesetter—and this number will vary from one machine to another, from as few as 88 characters to as many as 128. It may be that there are no "fonts" at all, but simply groupings of characters, sometimes called subsets of fonts, which are somehow accessed or "called in" when required. We shall get back to this subject subsequently, since it is relevant to a discussion of how you store and access the characters you may require for typesetting.

The concept of "set"

We turn now to one of the more difficult concepts in the field of typography. This has to do with what is called "set" or "set size." This concept is difficult to explain because there are, in reality, three different notions about set, and they are not at all the same. It is important that the reader understand each of these and the implications of each.

Originally, in hot metal (hand composition and linecasting, but not Monotype) *set* meant the way in which type was designed, and only that. In other words, to achieve a more condensed or a more expanded type face, type was often designed so that one type face would match another so far as "point size" or character height is concerned, but would offer a more expanded or more condensed appearance. And even for the Monotype system, the way you would come up with a condensed face would be to redefine the "em" so that instead of it being, say, ten points, for a ten-point "face," the em "set size" would be 9½ or even 9 points—not so far as height goes, but so far as *width* is concerned.

8 PT. (8D) $8^{1}/_{4}$ SET LINE M·1252
U.A. 90

WHEN JOBS HAVE TYPE SIZES FIXED
When jobs have type sizes fixed quickly margins of error can widen unless all determining calculations are based on factual rather than
When jobs have type sizes fixed quickly margins of error can widen unless all determining calculations are based on
ABCDEFGHIJKLMNOPQRST

If you are interested in page depth, and in the spacing between lines, you need to know that you are dealing, with, say, a ten-point face so far as character height goes—and whether or not the type is going to be leaded to eleven or twelve. If, however, you are interested to know how many characters will fit *across a line*, you will want to know that, for these purposes, the face is *nine set* as opposed to *ten set*. Consequently, we must bear in mind that an em space is not necessarily as wide as the point size of the type, but *only as wide as the set size.*

7D ON 8 PT. 7 SET LINE M·1232
U.A. 441

WHEN JOBS HAVE THEIR TYPE SIZES
When jobs have their type sizes fixed quickly the margins of error will widen unless all determining calculations are based on factual
When jobs have their type sizes fixed quickly margins of error will widen
ABCDEFGHIJKLMNOPQRSTU

It is likewise possible to design a type face so that its set is *wider* than its point size, and thus you would obtain an expanded (rather than a condensed) face. And so the first definition of *set* is related to the *design of the original type.* In the margin are two samples of type so designed.

A second definition of set has to do with the manner in which a *given design* of type is laid down across a line. It would be possible in Monotype, for example,

to cast a certain face on a somewhat wider body. In photocomposition it is usually possible to change the amount of escapement of each character—either in absolute units or in relative terms. In the former case, you might add or subtract, say, one unit to or from the width of each character's escapement. In the latter case you would add or subtract, say, one-tenth of that character's unit width.

To illustrate this we will compose three lines of type with different escapements. The phototypesetter we are using gives us the capability of reducing the escapement of each character in successive increments of one-half unit. We shall set the same line in a normal manner, and then with "minus one-half," "minus one," and "minus one and one-half" units.

Here is our sample line, set quad left to avoid justification.
Here is our sample line, set quad left to avoid justification.
Here is our sample line, set quad left to avoid justification.
Here is our sample line, set quad left to avoid justification.

So, as you can see, the *side bearings* of the characters have been reduced, but not the true widths of the characters themselves. Another way to accomplish a similar effect would be to subtract one-eighth and one-quarter set from the true set size of these same lines:

Here is our sample line, set quad left to avoid justification.
Here is our sample line, set quad left to avoid justification.
Here is our sample line, set quad left to avoid justification.

We could even call for a more extreme amount of white space reduction—say, one-half unit:

Here is our sample line, set quad left to avoid justification.

Or, we could *add* side bearing space. In this instance we will first show the normal line, and then add one-eighth, one-quarter, and one-half set:

Here is our sample line, set quad left to avoid justification.
Here is our sample line, set quad left to avoid justification.
Here is our sample line, set quad left to avoid justification.
Here is our sample line, set quad left to avoid justification.

Now there is a third definition of set which applies only to "third generation" or cathode-ray tube typesetters. It is possible, with CRT typesetters, as we shall explain subsequently, to play "electronic games" with character generation, and to distort the type characters from their original design, so that they are wider or narrower, or taller or shorter. It is possible to draw a character or series of characters on the face of a CRT tube and to compose them as if they were ten points high by nine points wide, or even 18 points high by six points wide. The result, so far as the image is concerned, is precisely as if the type had been designed to a different set (as in our first definition). The difference, however, is that the type

was not so designed, and is being created electronically, for this one setting, *as if* it had been designed in this fashion. Thus it is not the side bearings which are changed, but the proportions of the characters themselves. Here are two examples, set on a Fototronic CRT typesetter. The master size is 12 points, and the first and last line are "set" to true width—namely twelve set. The widest line illustrates expansion to equivalent 18 point width while retaining 12 point height, the width is decreased one point for each successive line to a minimum width value of 8 points. (Samples courtesy of Rocappi, Inc.)

abcdefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyz

Palatino

abcdefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyz

Times

Note, however, that in *all three cases*, it is "set"—however defined or applied —which determines the fit of characters within the given line and the number of characters which can be set across a given amount of space.

But even within a given "set," some type faces will permit the placement of more characters across a line than will others. This has to do with the relative proportionality of the face in question. Two ten-set type faces can have different *character counts*. A *character count* is an indication of the width of a lower-case alphabet. It assumes that each character is equally important in the flow of text, which is, however, not the case. But at least it does indicate to you that some type faces are wider than others. Here are some examples of different fonts or faces of type with their character count statistics. In all instances, however, the faces shown below were nevertheless designed to the same "set."

10 point—1½ point leaded 2.78 characters to 1 pica

Simplicity is a very important feature in typography, because it produces the direct appeal. It is this element

ABCDEFGHIJKLMNOPQRSTUVWXYZ $1234567890

ABCDEFGHIJKLMNOPQRSTUVWXYZ SIMPLICITY IS A VERY

10 point medium—1½ point leaded 3.02 characters to 1 pica

Simplicity is a very important feature in typography, because it produces the direct appeal. It is this element of simplicity, which makes for easier reading as well as better

ABCDEFGHIJKLMNOPQRSTUVWXYZ $1234567890

With this background about the characteristics of type it is now possible to discuss, first of all, the major hot metal typesetting "systems," and, later on, photocomposition devices.

Review Questions:

1. What is the characteristic shape of foundry type?
2. In what situations would it be advisable to mortise hot metal type characters?
3. How is point size defined?
4. What must happen to the "x" or "z" height of a character if its ascender or descender is extended?
5. How would you define "leading," and what is the origin of the term?
6. What is the difference between primary and secondary leading?
7. Which is the longer unit of measure: a point or a pica? How many points to the inch? How many picas to the inch?
8. How wide is an em space?
9. If you were setting 18-point type, how wide would an em be?
10. What are some of the advantages in the use of a relative system of type measurement?
11. What is roman type? What is sans serif type? What are serifs?
12. Name some of the categories into which type designs might be classified?
13. Is the concept of "set size" important in terms of the vertical, or the horizontal dimensions of a page?
14. Explain the three meanings of "set."
15. Which of the above three meanings would correspond to negative or positive letterspacing?

4

The Linecasting Machine And its Relation to the Production Process

IT IS NOT OUR purpose to devote much time to the history of typesetting. We want to concentrate virtually exclusively upon photocomposition systems and devices—especially those which are driven or controlled electronically—since this is the way "modern-day" composition is being performed. We would like the reader to understand, however, that our concentration upon these so-called "modern" devices, or the "new technology," does not necessarily imply that all older devices and methods are undesirable, or outdated, or are no longer relevant for any purpose or function. The fact is that older techniques are still very useful in some—perhaps many—situations. And such techniques have an art form all their own, so that many people will continue to produce books and other printed pieces by such methods. In fact, there are a good many "private presses" in the United States, and no doubt elsewhere in the world, where books are composed elegantly, usually by hand type, on small job presses (or even "wine" press models), and on ancient papers. Wood type is often used, too, just as are wood block engravings.

But new composition techniques are also available, and they offer many economies, as well as the advantage of timeliness. They also hold forth their own art form possibilities. In writing of these new techniques we do not intend to be judgmental about the old, but have in mind broadening the reader's horizons for whatever purposes he may have in mind in the reading of this book. Nevertheless, we need to lay some foundation as it relates to the nature and function of these older hot metal processes, for otherwise the reader will not understand or appreciate what the new techniques are, do, and imply.

The linecaster

Since the turn of the century and until, perhaps, about 1970, most composition has been produced, at least in the United States, by the use of linecasting machines, although some quantity of type—ever decreasing even prior to the advent of photocomposition—was set by Monotype.

The Linecasting Machine

Ottmar Mergenthaler's Linotype was first introduced in the mid-1880s, the patent having been granted in 1886. Mergenthaler was not the only person to research the possibility of developing an automatic typesetting device. A chronological listing of "Type-setting Machines and Ancillary Equipment, 1822–1925" appears in *The Journal of Typographic Research*[1] for July 1967, and it includes 173 such devices! Of these, 57 preceded Mergenthaler's patents. The list begins with two inventions by Dr. William Church (an American who had settled in Birmingham, England) in 1822. The tempo of developing technology is interesting in itself: 1839—Kiegl (Hungarian); 1840—Clay and Rosenborg (English); 1843—Gaubert (English); 1840—Pianotyp by Young and Delcambre (English); 1843—Mazzini (English); 1846—Tschulik (Austrian); 1846—Letter Printing Telegraph, sponsored by *Scientific American* magazine; 1849—Tacheotype (Danish); 1850—Mitchell (American); 1853—Jones (American); 1854—Beaumont (American); 1854—DeSimoncourte (English); 1854—Poole (English); 1854—Thomas (American); 1855—Loughborough (American); 1856—Koenig (American); 1857—Alden (American); 1857—Hattersley (English); 1857— Houston (American); 1859—Gilmer (American); 1860 Felt (American); 1862—Brown (American); 1862—Fraser (English); 1862—Ray and Mackay (American); 1866—Baer (American); 1867—Corey and Harper (American); 1867—Kniaghininski (Russian); 1867—Mackie (English); 1868—Foster (American); 1869—Kasterbein (French); 1870—Klees (American); 1871—Hooker (English); 1871—Plunket (American); 1872—Westcott (American); 1873—Filmer (American); 1873—Paige (American); 1874—Goodale (American); 1875—Burr (American); 1876 Stone (American); 1877—Farnham (American); 1878-Wicks (English); 1880—McClintock (American); 1880—Fleming (American); 1880—Thorne (American); 1882—McMillan (American); 1883—Clephane (American); 1884—Mergenthaler Band Machine #1; 1885—Dow (American); 1885—Lagerman (Swedish); 1885—Lyman (American); 1885—Mergenthaler Band Machine #2 and Mergenthaler circulating matrix machine (American); 1885—Mergenthaler "Blower" Linotype (replaced by the Square-based Linotype in 1889.)

But this was by no means the end of the road. In addition to the Lanston invention, in 1887, there were many others—more than one hundred, most of which never truly became products and were never, in fact, used in a production environment.

All of this effort clearly demonstrates the old adage that necessity is the mother of invention. People were not happy with the methods of setting type which preceded the Linotype and the Monotype, and they still tried, even after the introduction of these products, to come up with better solutions.

It was Linotype and Monotype that survived, plus the Intertype linecasters which came along after the basic Mergenthaler patent had expired. Nowadays, when people talk about "Linotype" or the "Linotype process" they often use the term in a generic sense to embrace Intertype as well, since the principles of the Intertype machine are virtually identical.

1. *The Journal of Typographic Research*, Vol. 1, No. 3, July, 1967, 245–274. Article by Richard E. Huss. Published at that time by The Press of Western Reserve University, 2029 Adelbert Road, Cleveland, Ohio, 44106. *See also*, Huss's book on this subject, *The Development of Printers' Mechanical Typesetting Methods, 1822–1925*. University of Virginia Press, 1973.

At this point we shall devote our attention to the linecaster. The Mergenthaler Linotype Company sold approximately 74,000 such typesetters from 1893 through 1968. Between 1893 and 1900 almost 4,500 were delivered, which averaged approximately 645 per year. From 1901 through 1906 shipments increased to more than 800 yearly, and from 1907 through 1916 approximately a thousand a year were delivered. The rate of shipments went up to more than 1,700 per year from 1917 through 1922, and then fell to about 1,100 yearly until 1927. In the three years, 1927–1929, they averaged 1,666 per year, dropped off to 710 between 1930 and 1936, moved up to 833 from 1937–1942, declined to 783 between 1943 and 1948, moved back up to 1,250 per year in the 1949–52 period, and then averaged only 510 per year from 1953 through 1959. However, in the meantime, Mergenthaler's popular "Comet" was released (for which statistics were kept separately) and from 1951 through 1958 something like 330 Comets were delivered each year.

But Mergenthaler's sales do not tell the entire story. Intertype began to sell in 1913, and by 1957 they had distributed some 27,000 units, and this figure rose to a total of about 38,000 by the late 1960s. In fact, Intertype's sales from 1950 on seem to have equalled or exceeded those of Mergenthaler. In 1956, for example, 1,150 units were shipped.

Thus it may be seen that from the period of the invention of the linecaster until the point where they were no longer made in the United States (say about 1970) more than 100,000 linecasters were sold within the U.S. and Canada, or, in some cases, shipped abroad from U.S. factories. Even allowing for obsolescence, surely more than 35,000 linecasters stood on composing room floors in the U.S. in the late 1960s, available to be used in production.

From the year 1886, when the first Linotype was installed in *The New York Tribune*, until 1893, and on even until about 1900, there was gradual accommodation to the new technology. Whereas a skilled hand compositor was said to be able to compose about 2,000 characters per hour, the average output of the Linotype was estimated at about 6,000 letters, and perhaps more important, the task of "distribution of type"—so frequently associated with hand composition—was no longer required. The Linotype's circulating matrix provided an ever-ready source for new type images, and the slugs of type cast from the assembled matrices, after their use for printing, were recycled via the "hell box" into the melting pot.

There were other advantages to linecasting besides the avoidance of distribution. Lines as slugs were easier to handle, and blocks of type were less likely to "pi"—that is, to become mixed up. Justification was much simpler—and this accounts for a great deal of the increase in the amount of composition set per person—since the "spacing in" and "spacing out" techniques for justification and tight setting necessary in hand composition were totally eliminated. No longer did you have to substitute, experimentally, smaller or larger fixed spaces (such as the three-em, four-em, or five-em—that is, one-third, one-fourth or one-fifth em spaces)—between words until the line was tight enough but not too tight, and

the spaces between words were more or less evenly distributed. Variable spaces, created by spacebands, took the place of fixed spaces of varying widths, and all the operator had to do was to *bring the line within justification range*. The actual justification was in fact accomplished by spaceband expansion.

And so it is the production technique which the linecaster brought about—in some instances not too much different from that which went before, when type was composed by hand—that provides the starting point for our examination.

Principles of the linecaster

It is not our function here to provide a dissertation on the linecaster, but it will be helpful to set forth a few general notions about it, and perhaps the best way is to quote from Mergenthaler's *Linotype Machine Principles*—the official manual—copyright 1940.

The Linotype is not a type-setting machine. No types are used in it. It composes with matrices—small brass units having characters indented in the edges—hence the name "matrix." These matrices are assembled into justified lines. From the matrix line the Linotype automatically casts a solid bar, or line, of type. This bar is known as the Linotype slug. It is ready for use when it leaves the machine.

The Linotype has four major divisions:

1. The Magazines which contain the matrices.—They represent type cases. Because every matrix circulates automatically back to its place in the magazine as soon as it has served in a line of composition, a font of matrices is small in number as compared with a font of type. A magazine is so compact and light that the operator can handle it without exertion, and produce a variety of composition simply by changing magazines.

2. The Keyboard and its related parts.—This controls the release of matrices from the magazine in the order in which the characters are desired. The Linotype operator, from his seat at the keyboard, has complete control of every function of the machine. His duties are limited merely to operating the keyboard keys—justification and distribution are mechanically automatic.

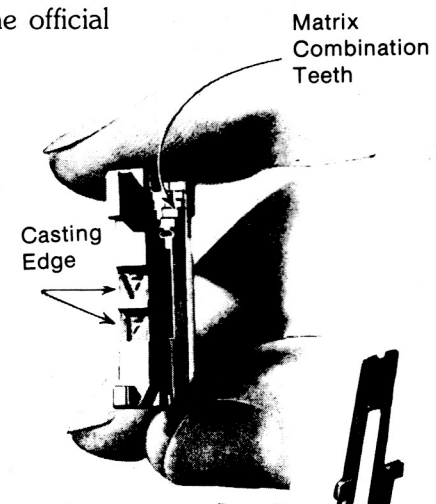
3. The Casting Mechanism.—The division of the machine makes the Linotype-equipped printer his own type founder. The justified line of matrices is presented automatically to the casting mechanism, molten type metal is forced into the indented characters on the edges of the matrices, and the cast line, a single unit with a new type-face, is delivered to galley on the machine, precisely trimmed and ready to go into the form.

4. The Distributing Mechanism.—When a line of matrices has served for casting the line of type, it is lifted automatically and carried to the top of the magazines, where, by a simple though ingenious system, each matrix is delivered to its proper place in the magazine and is ready to serve again. Thus, in the Linotype-equipped shop there is no distribution of type.

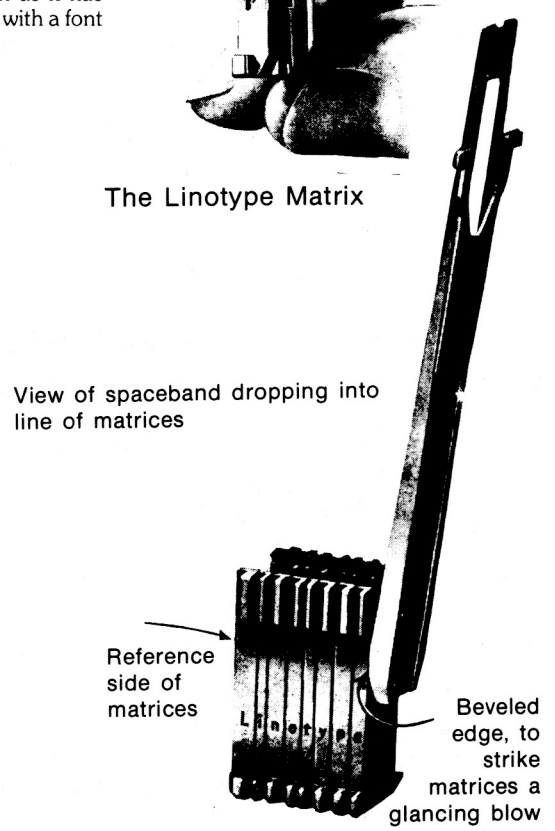
Note the teeth at the top. Each character has a different set of teeth. These guide it back to its proper place for repeated use. . . .

The spaceband consists merely of two thin wedge-shaped pieces of metal. They taper in opposite directions so that the outside faces are always parallel.

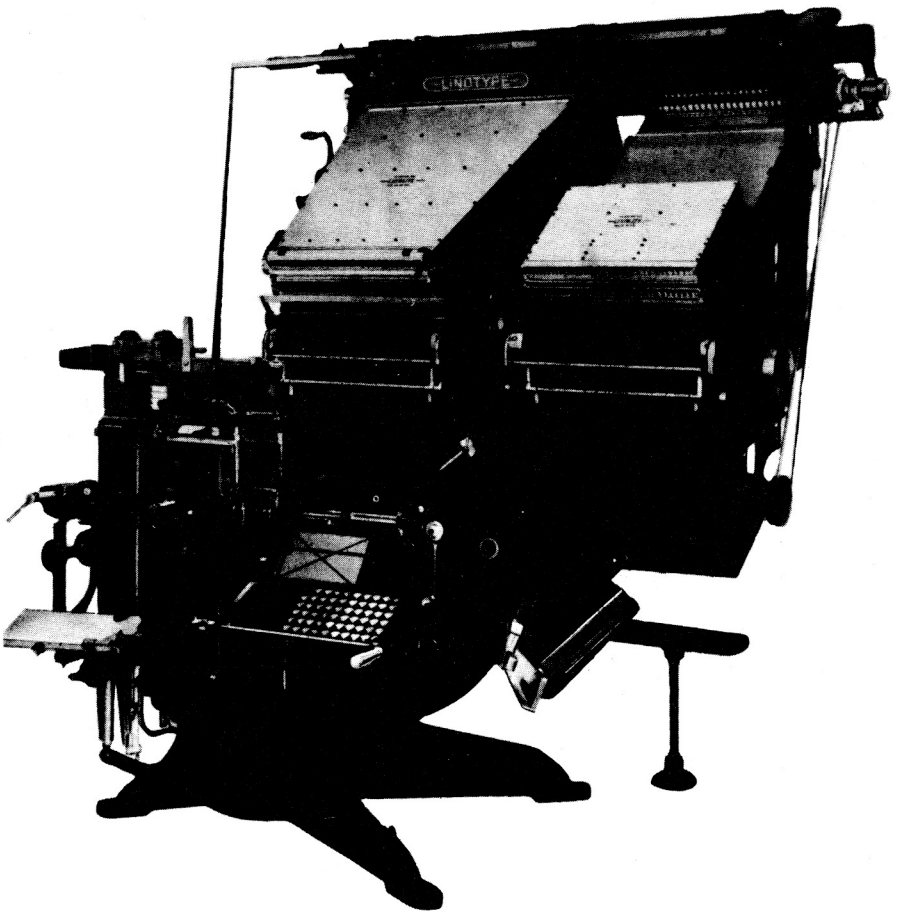
To the right are pictures of the Linotype matrix, showing the intaglio image in the casting edge and the teeth that guide it back into its channels in its magazine. Also illustrated is the way in which a spaceband falls alongside the matrices.

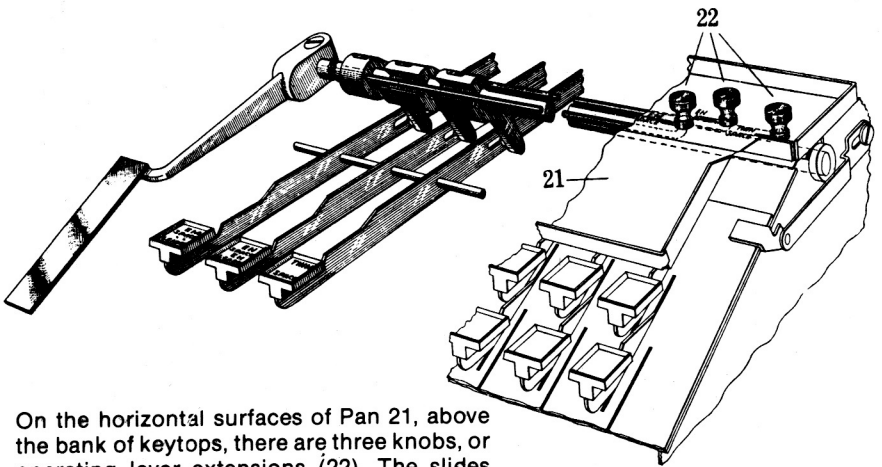


The Linotype Matrix

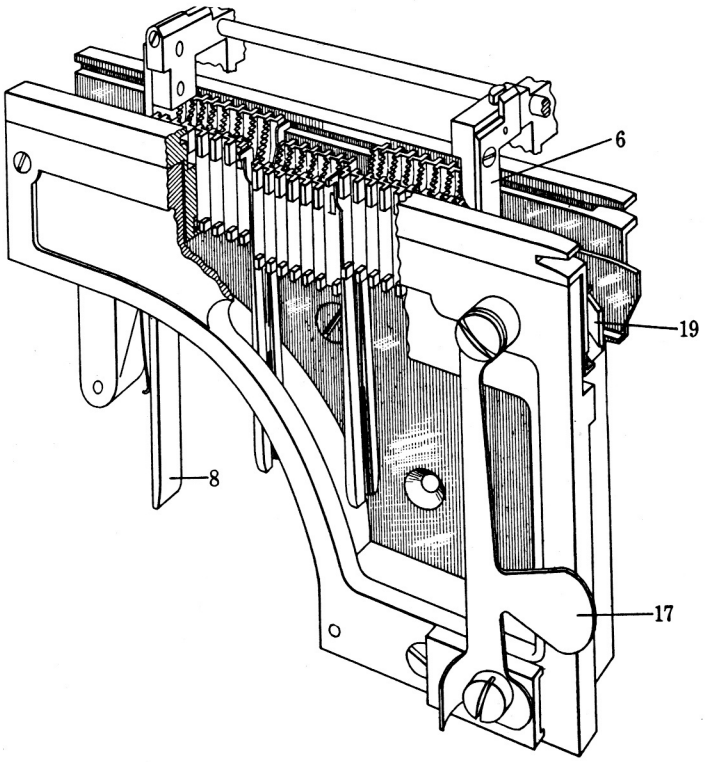


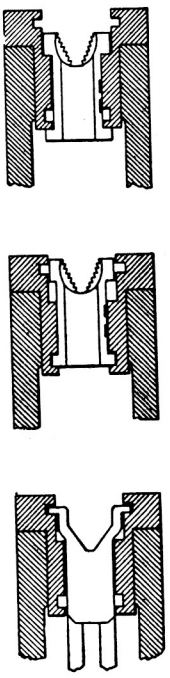
Model 30 Linotype
as viewed from the front

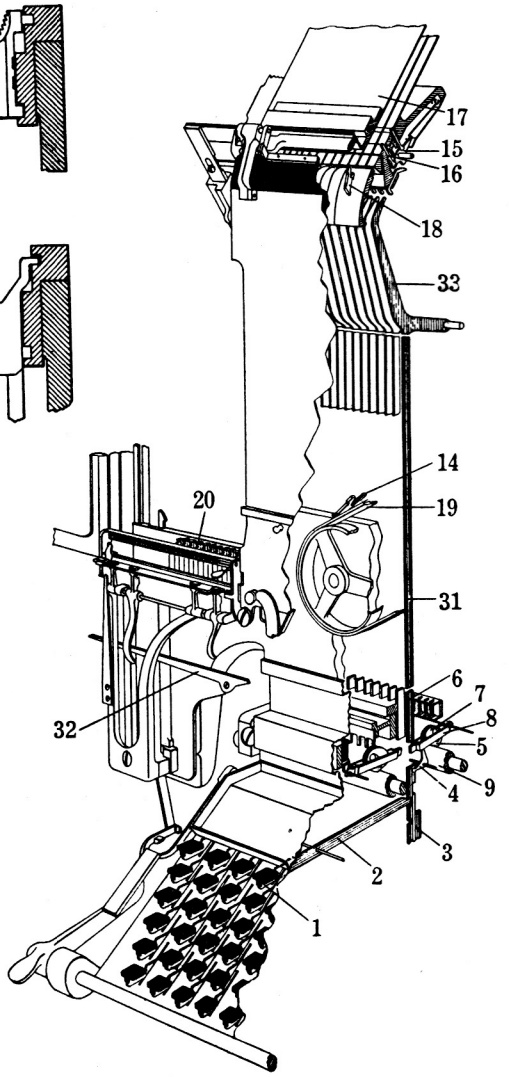




On the horizontal surfaces of Pan 21, above the bank of keytops, there are three knobs, or operating lever extensions (22). The slides which are operated from these knobs cause the operation of the spaceband key to release a spaceband and any one or any combination of the em, en or thin spaces at the same time. When these knobs are all moved to the right as shown above, the spaceband lever operates normally. The combination of a fixed space or spaces with the justifying space is advantageous in many kinds of composition, such as centered lines, etc.







View of the delivery channel with a portion of the channel cut away to show the matrices and spacebands in different positions. The channel is simply a passageway for the matrices and spacebands in the same position in which they are received from the assembling elevator. The sectional views at the upper right show how the matrices and spacebands are supported while passing through the delivery channel.

At right: Perspective diagram of the assembling mechanism of the Linotype. *1* is the keybutton; *2*, keylever; *3* is the key weight, called the keyboard bar; *4*, the trigger; *5*, keyboard cam; *6*, keyboard rod, or key reed; *7*, keyboard stop cam pin; *8*, keyboard cam yoke; *14*, matrix on assembler belt; *17*, magazine; *18*, matrix; *19*, matrix delivery belt; *20*, assembling elevator; *32*, spaceband buffer finger; *33*, escapement lever.

The nature of the linecasting production process

Manuscript is received in the composing room, where it is "copy cut" according to the type faces, sizes and measures of type to be composed. Usually the manuscript is marked by the copy editor with symbols to indicate the desired measure, faces and spacing, but in some instances markup is left to the discretion

of the composing room mark-up man or the copy cutter. It must be remembered that a Linotype machine can compose only to one measure without a manual adjustment of the pre-set vise jaws. Lines of up to 30 picas can be cast on all machines, and some are built to accommodate lines of up to 42 picas. Machines requiring a still wider measure can sometimes create "butted" slugs—which means, in effect, that two lines can be composed and laid side by side in an interlocking fashion.

On a typical linecaster, one magazine of 91 channels is available at one time. Such a magazine weighs between 50 and 60 pounds when filled with matrices. One magazine contains one font of type of a particular point size (from four to 60 points, with text machines usually setting up to 18 points). Since duplexed matrices are generally used, it is thus possible to compose roman and italic or roman and bold, setting to a certain measure, in a certain point size, without the need to change magazines or reset vise jaws.[1] Some linecasters have "mixing" capabilities. Mixers have multiple distributors so that circulating matrices can be returned to the appropriate magazines. Other models may permit the mounting of several magazines simultaneously, although only one can be used at a time. There are also models which offer side or auxiliary magazines to enhance the available character repertoire.

1. *See* illustration of the assembly of matrices to access two different type faces—up and down the "rail" on the preceding page.

Copy Cutting.—Generally speaking, then, several operators will be involved in the production of one job, depending upon the type faces, sizes and measures required. If a book were being set, consisting of body text, footnotes, extract, and some display material, the manuscript would either be cut into sections, or routed from one operator to another so that each could key relevant portions. Sometimes several machines are set up with identical magazines so that more than one operator can compose text matter on the same job. Nevertheless, except in the newspaper industry, where most of the copy consists of news text or ad "agate" to be set to a given face, size and measure, the number of persons who can share the task of composing the same job at the same time is generally quite limited. Matrices and magazines have been expensive. So were the linecasters themselves. No owner wants to pay for idle equipment. Consequently, the effective management of men and machines in hot metal shops is complicated, and one of the factors that often slows down production on a given job is the problem of the lack of availability (within the shop) of the required fonts. For example, one shop may have one or two wide-measure machines, one or two "mixers," and three or four 30-pica models. It might also own only two magazines containing ten-point Times Roman, two containing ten point Caledonia, one with eight point Times, and several magazines each with a particular size of display text.

While some operators are capable of setting complicated text, others cannot be trusted to compose difficult material. The reader can readily understand that the task of supervision and scheduling is indeed complicated—more so, of course, in a trade shop performing a wide variety of jobs than in a production shop associated with a newspaper, for example, where experience develops a routine and where work flow is predictable from day to day.

The work flow.—The linecaster operator composes type slugs which are then placed in galleys. Proofs are pulled, often by an apprentice or a proofpress boy. Each galley is identified, and the type is laid out sequentially—although, for proofing purposes it often has not yet been assembled into its final order. In other words, it may be necessary to proofread all the extract material assembled into one galley, footnotes from another, text from a third, and "display" from still a fourth. On the other hand, the various elements may be combined into their proper order by hand compositors or "bank men" before galley proofs are supplied.

Errors are marked by proofreaders with their own special annotations, and it is then necessary to compose corrections. This means that the galleys have to go back to operators—not necessarily the same ones (but often!)—who sit at the machines on which the proper magazines are currently mounted (or can be mounted), adjusted to the proper measure. It often happens, of course, that the setting of corrections is delayed until machines or operators can be freed up from other assignments. But finally the necessary slugs will be reset. Often corrections require setting not merely the line that contains the error, but down to the end of the paragraph, since the substitution of even one letter for another can sometimes result in the rejustification of a line which, in turn, can have an impact upon the next, and so on until the change works itself out or the paragraph comes to an end.

Usually these corrections are proofread before they are incorporated into text, and, of course, new errors can crop up, often in text that has been correctly set the first time, but which has had to be reset because of justification problems. In this case, corrections are made to corrections, and these are usually assembled again for proofreading. Finally the proofreaders are satisfied and the corrected matter is put together with the original uncorrected text. This means removing some line slugs and substituting others. There is, of course, the possibility that the wrong slugs may be removed or replaced, or that the galley itself might be tipped and a number of lines thrown out of sequence. Therefore another excursion into the proofreading department is in order, to verify that the corrections have been inserted into the proper position, and no new errors have been inadvertently created.

Now, if not before, is the time to assemble all of the various typographic ingredients—the extract, the display, the footnotes—as much as possible in the proper order. This permits new galley proofs to be pulled, which may go out again to the proofroom, and then, finally—in the case of outside work—to the customer.

Remember that we are not yet "into" pages. Perhaps the publisher or editor already has some approximate idea of the length of his job. Perhaps the initial manuscript has been written to some copyfitting formula. Perhaps it was typed with a predetermined number of lines per page and characters per line, in the belief that, when set in type, a certain result in terms of column depth, number of

pages, or even the number of signatures, can be predicted. It is still virtually certain that "pagination" problems will present themselves.

But at this point it is now easier to see more precisely what the length of the material will be—not to the exact number of lines, of course, because page make-up poses its own problems. Editing may therefore be necessary to achieve better fit. The publisher or editor may discover the unfortunate fact that some operators set "tighter" than others, and projections as to the length of the material may have been thrown off as a result.

It is also likely that the editor, or, more probably the author, will introduce further changes—for he, too, will see the galleys, and, indeed, in the case of a book or an article, he will have been looking forward to the moment when his words are memorialized into type. As we all know, the trauma of seeing what you have written now set in type cannot help but induce a fervent desire for last-minute changes. A word here or there, the turning of a more felicitous phrase, a degree more precision, a *bit* more emphasis, an aside that did not occur to the author before but now strikes him as singularly apt, and the addition of a few facts that have arrived on the scene since he dispatched the manuscript to the editor some months before: All of these produce "author's alterations" and may require a substantial amount of resetting.

We shall not trace the process of introducing these changes, of proofreading them, or checking to see that they are property inserted, of dividing the galleys into pages, of then inserting the running heads and folios, of positioning the illustrations, of resetting lines so that they will make proper reference to the illustrative material, of assembling the book and then waiting until an index—which refers to the page numbers—can now be created, read and corrected. Obviously the process goes on and on.

But this is the procedure which has evolved over the years, and much of it has been determined or conditioned by the technology of the linecasting typesetting machine.

We will have occasion, shortly, to see how many of these steps can be eliminated or greatly simplified through photocomp. Much more material can be composed sequentially. Type faces can be mixed, line lengths can be changed. Display, extract, and even footnotes can be composed as encountered. White space can be assigned for illustrations. And when corrections are required, one machine is likely to have available, at one time, all of the fonts, sizes and capabilities desired. Moreover, generally speaking, the quality of the composition—that is, the justification of the lines themselves—may be entirely operator-independent.

New processes present problems, of course. The publisher-customer is familiar with old ways, just as is the supplier. Sometimes it is easier for new users to adopt new procedures than for old ones to change. Nevertheless, today linecasting is becoming much more of a rarity.

Other peculiarities resulting from the use of the linecaster

Type is composed by circulating matrices. It is not rare for matrices from the wrong font to find their way into a particular magazine. With circulating matrices (and therefore with a number of replications of the same matrix stored in a given channel so that there will always be a desired character on hand —particularly the common lower case letters such as *e* or *t*) it is also possible that some of the mats will become defective. Thus it may happen that certain individual letters—but not all of them—may become imperfect or out of alignment. Moreover, mats may fall into the assembly area out of sequence, so that there are often machine-generated errors as well as those created by operators.

Another common problem is the failure of the man on the "bank" who is charged with the responsibility of removing slugs which were incorrectly set—or have been revised—to remove the *proper* ones and to substitute the intended slugs in the proper sequence. Of course the compositor is accustomed to reading his type upside down and backwards, but for some reason such mistakes do occur more frequently than we would like to believe.

This, then, summarizes our presentation of the essence of the linecasting process which has for so many years dominated the field of composition in the English-speaking world and which has left its rich heritage to guide us into the future, where new techniques and new relationships are made possible by the miracles of modern electronics.

Review Questions:

1. Describe the process by which a line of type is cast on a Linotype machine.
2. Sometimes we have taken some pains to use the term "linecaster" rather than "Linotype." Why is this?
3. What are the essential differences between a linecaster and Monotype?
4. How does a linecaster perform justification?
5. How does the linecaster operator or "keyboarder" know when he is within justification range?
6. What is the function of the spaceband? How does this differ from a "fixed space"?
7. Briefly describe the basic production flow of a manuscript submitted for composition to a finished product in a Linotype shop.
8. What are the advantages of the linecasting system over the handsetting of type?
9. How are corrections handled on Linotype galleys?
10. What are some of the more cumbersome aspects of the Linotype process?
11. How many characters are available at one time on a typical linecaster? Is there more than one matrix for each character? Is there more than one character for each matrix? How would you decide how many matrices to provide for a given character? Would this differ according to the length of the line you were setting?
12. How do you change the measure (line length) on a linecaster?

13. How do you change the point size on a linecaster?
14. How do you change the type face on a linecaster?
15. What does "up the rail" mean? What is a "mixer"? What is the maximum line length you can set on a linecaster? What are "butted slugs?"
16. Why do you suppose that the linecaster keyboard was not arranged to resemble the typewriter keyboard, both with respect to the use of "upshift" and "downshift" keys, and, in general, with respect to the keyboard arrangement? Why not use a "qwerty" keyboard instead of an "etaoin shrdlu" keyboard?
17. How would you insert "pi mats" into the text of your composition?
18. How would you set a "run-around" if you wanted a half-column cut?
19. How would you set tabular material on a linecaster?
20. What is an "automatic quadder"? Is there an "automatic leadering" function on a linecaster?
21. In your judgment, would you expect the type from a linecaster to be more uniform in appearance than the type from a photocomposer? What factors would influence the question of uniformity?

5

Monotype Composition: The Process and its Contribution

AS WE STATED in the preceding chapter, the Monotype method of composition was patented in 1887, and was first exhibited in 1889. Tolbert Lanston (1844–1913) was the inventor of this process. A sergeant in the U.S. Civil War, he subsequently became a civil service clerk in the U.S. Patent Office. Although he took a law degree in his spare time, he devoted his leisure to "gadgetry," inventing an adjustable horseshoe, a mail bag lock, and an adding machine. When he saw the Hollerith Tabulator, presumably at the Census Bureau,[1] the idea occurred to him to develop a typesetting machine driven by coded paper tape.

1. *See* our discussion of the Hollerith card in our treatment of Computers in Chapter 10.

After the prototype was displayed at the Columbian Exposition in Chicago, production of the first 50 machines was undertaken under contract with the Philadelphia firm of Sellers & Co.

Originally, the "caster" was a die stamping device, which reproduced single types from cold strips of type-high lead, with a character repertoire, in terms of the stamping case, of 225 characters. Shortly it was modified to become a metal caster, with a font of only 132 matrices.

Four of these machines were shipped to England, and during the Atlantic crossing two of Lanston's associates met up with the Earl of Dunraven, who formed a syndicate virtually on the spot to purchase (for one million dollars) British rights to the patent. This development solved the American company's financial crisis and provided money for further development. Thus the Lanston Monotype Corporation was founded in England in 1897.

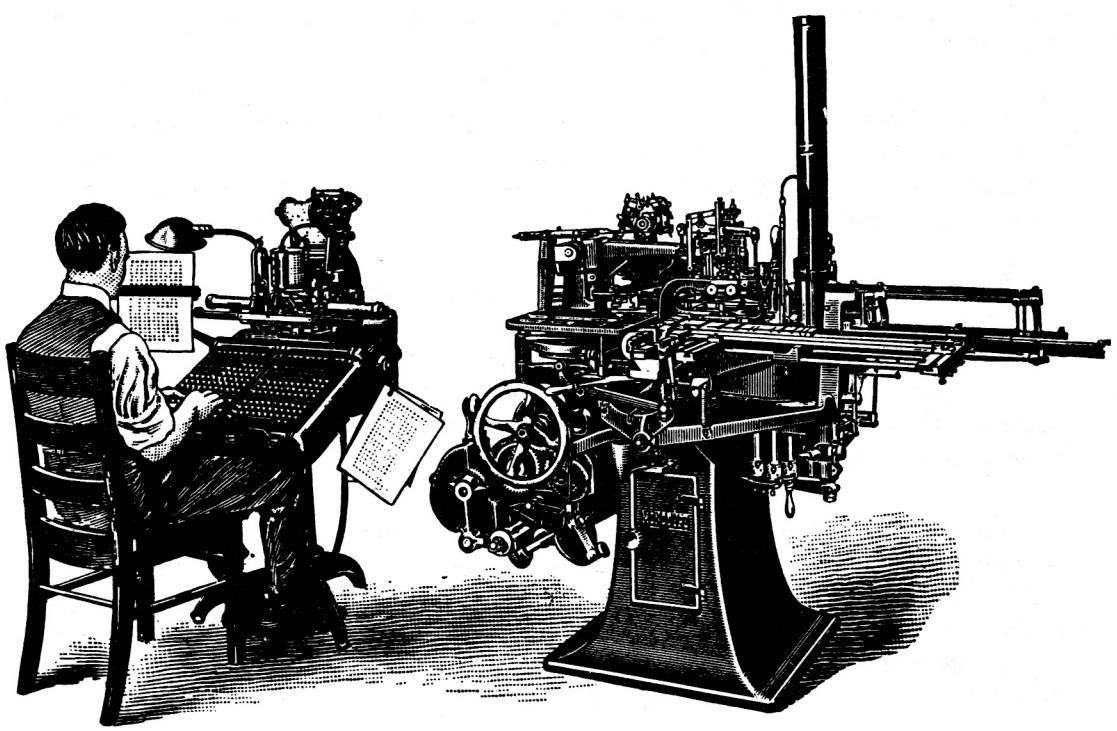
The British company manufactured component parts, including molds and matrices. It also marketed the product throughout the world, outside of North and South America, but it did not manufacture keyboards or casters until after World War I. Meanwhile, substantial improvements were achieved in the design of the machine: a justification wedge was added, and the matrix case was restored to its full character set of 225 positions.

The British company established its principal "works" at Salfords, about twenty miles south of London. The company began to make significant strides after World War I. During that war its facilities were devoted to the manufacture of machine gun components, but in 1922 work was commenced on the actual fabrication at Salfords of complete keyboards and casters. Exclusively British products were first marketed in 1927.

From that point on the paths of the two companies diverged increasingly. There was no common ownership of shares, no centralization of decision-making. Monotype (Ltd.) began its important work in the field of "typecutting" following the appointment of Stanley Morison as typographical advisor in 1922.

World War II again converted the Salfords Works into a munitions factory. When the war was over the company was literally overwhelmed by the demand for its products. This was not at all the case in the United States, with Lanston, where linecasting had already made much greater inroads. The decline of the American company was already obvious in the mid-1950s and in the 1960s it was absorbed by other financial interests and then ultimately it was discontinued.

We have devoted space to this brief bit of history to make it clear that Monotype today is a British company, and that the principal market for its hot metal products is international. The flexibility of the Monotype system readily lends itself to the setting of foreign languages—even, with casting machine modifications, Arabic and Hebrew.



THE COMPLETE MONOTYPE

Principles of operation

It is worth spending time to understand the Monotype system because it represents the first application of *counting logic* to the typesetting industry. As we have already observed, there is no need for the compositor to know anything specific about the widths of various proportionally-spaced characters so long as the physical assembly of the types themselves is done manually (as with hand composition) or by the operator himself, observing the process of line filling which takes place in the assembly of the linecaster. But the separation of the keyboard from the casting machine in the Monotype system made it necessary, as we have mentioned several times, to provide the keyboarder with a counting mechanism. This brought about the adoption of a "relative unit" system, and a mechanical means whereby each stroke of a counter is incremented to show how much of the line remains.

The keyboard.—It is the function of the keyboard to perforate holes in a paper ribbon which is subsequently mounted onto the casting machine in order to "drive" it. Holes in the tape indicate which character positions are desired, or, more precisely, which location in the matrix case is to be accessed during the casting operation.[1] It is also the function of the keyboard to increment the "Unit Wheel" and to provide a pointer which will enable the operator to key, at the end of each line, the value of the interword space for that particular line. Since the paper ribbon is read backwards by the casting machine, the interword space widths are the first fact that the machine encounters, and these values are then used for the spaces for the line in question.[2]

1. Later, "Monomatic" models used a different principle to relate the ribbon to the mat case locations.

2. *The Monotype System*, Philadelphia, Pa., Lanston Monotype Company, 1916, copyright 1912.

> Speaking broadly, the counting mechanism includes *(a)* the unit registering mechanism, which measures the width of each character as struck (in units of the set of its face), and adds this number of units to the sum of the units of the characters preceding it in the line, in order that the counting mechanism may indicate the amount of space required to complete the line; *(b)* the mechanism for counting the justifying spaces, and *(c)* the calculating mechanism which determines, after the line is completed, the amount that must be added to each justifying space to spread the total amount the line is short over all the justifying spaces in the line. It will be clearer, however, to consider now only *(a)* and *(b)*; that is, the mechanism for measuring the number of units the line is short (after the last character has been struck) and the mechanism for counting the number of justifying spaces in the line to which the amount required for justification must be added.
>
> ****
>
> Consider first that portion of the counting mechanism that registers the unit width of the characters as their Keys are struck. Reduced to its simplest terms, this consists of the Unit Wheel (a35KB1) [in the illustration] and an escapement to regulate the amount the Wheel revolves when the key is struck. The Unit Wheel, a gear with 162 teeth, is urged to revolve contraclockwise by its Driving Rack, which is forced to the left by the air pressure in its cylinder (36KB1) acting upon the Driving Rack Piston. This Rack drives the Unit Wheel by a Pinion on the Unit Wheel Shaft (a35KB2).
>
> The Unit Wheel Pawl (a38KB1) is seated in the Unit Wheel, locking it and preventing it from rotating except when a key is depressed. The Pawl then lifts and the Wheel (driven by its Rack) revolves contraclockwise until as many of its teeth have passed the Pawl as there are units in the character struck.

Identification of Symbols

Paper Guide X45KC
Paper Spool X15KC
Paper Spool Shaft Xa16KC
Paper Feed Wheel a13KC1
Paper Tower Punch Guide Index Plate 18KC30
Justifying Scale Pinion Stud a13KB5
Justifying Scale 10KB1
Justifying Scale Pointer Rack a14KB3
Justifying Scale Pointer a14KB1
Paper Tower Housing Punch Lock Knob 18KC11
Piston Block Valve Handle 29KC17
Justifying Scale Weight 69KB1
Unit Rack Lever 73KB1
Unit Wheel Shaft a35KB2
Unit Wheel Driving Cylinder (left) .36KB1
Em Rack Slide a5KB1
Em Rack Stop Pointer 6KB3
Em Rack Stop Rack Adjusting Screw Head a8KB2
Em Rack Stop Pointer 6KB3
Unit Rack Slide a29KB1
Space Cut Out Operating Rod Head 16KA5
Keybank Button (137) 7KA6
Paper Feed Release Plate Link... b8KC2
Paper Ribbon (indicated by leader dots)
Paper Feed Release Plate Knob (knurled) 13KC10
Punch (29) 13KC1
Punch Bars (10) [also a32KC2 (11) and a33KC3 (12)] a33KC1
Unit Rack Stop Bar Base a32KB3
Paper Feed Rod 9KC1
Em Rack Stop X6KB
Em Scale Holder a9KB2
Valve Returning Rock Shaft Operating Arm Rod Head 14KA7
Bell Hammer 2KB1
Bell a1KB1
Unit Wheel Pawl Spring a31KB1
Unit Indicator a25KB1
Line Counter a23KB1
Restoring Rocker Arm Link Lever 24KB4
Unit Wheel Pawl a38KB1
Unit Wheel a35KB1
Em Rack 4KB1
Bell Trip Lever 3KB1
Em Rack Pointer a4KB3
Em Scale Clip a9KB5
Em Scale a9KB1
Keybank (right) 7KA1
Unit Rack Stop (9) [also 31KB2 (9)]31KB1
Unit Rack c26KB1

The Unit Rack (C26KB1) is the second member of the Unit Wheel escapement, the Pawl being the first. When a Key is depressed, the Rack moves up and engages the Wheel, and after the Rack is fully seated in the Wheel, the Pawl lifts, permitting the Unit Wheel to revolve and drive the Unit Rack to the right.

The Unit-rack stops (31KB1) complete this mechanism for measuring the number of units in each character struck, for when a Key is depressed, one of these stops rises in the path of the Unit Rack and stops its movement to the right and consequently the rotation of the Unit Wheel.

The Unit Indicator (a25KB1) enables the operator to tell at a glance the number of spaces the Wheel must revolve to seat the right tooth of the Pawl in a graduated space.

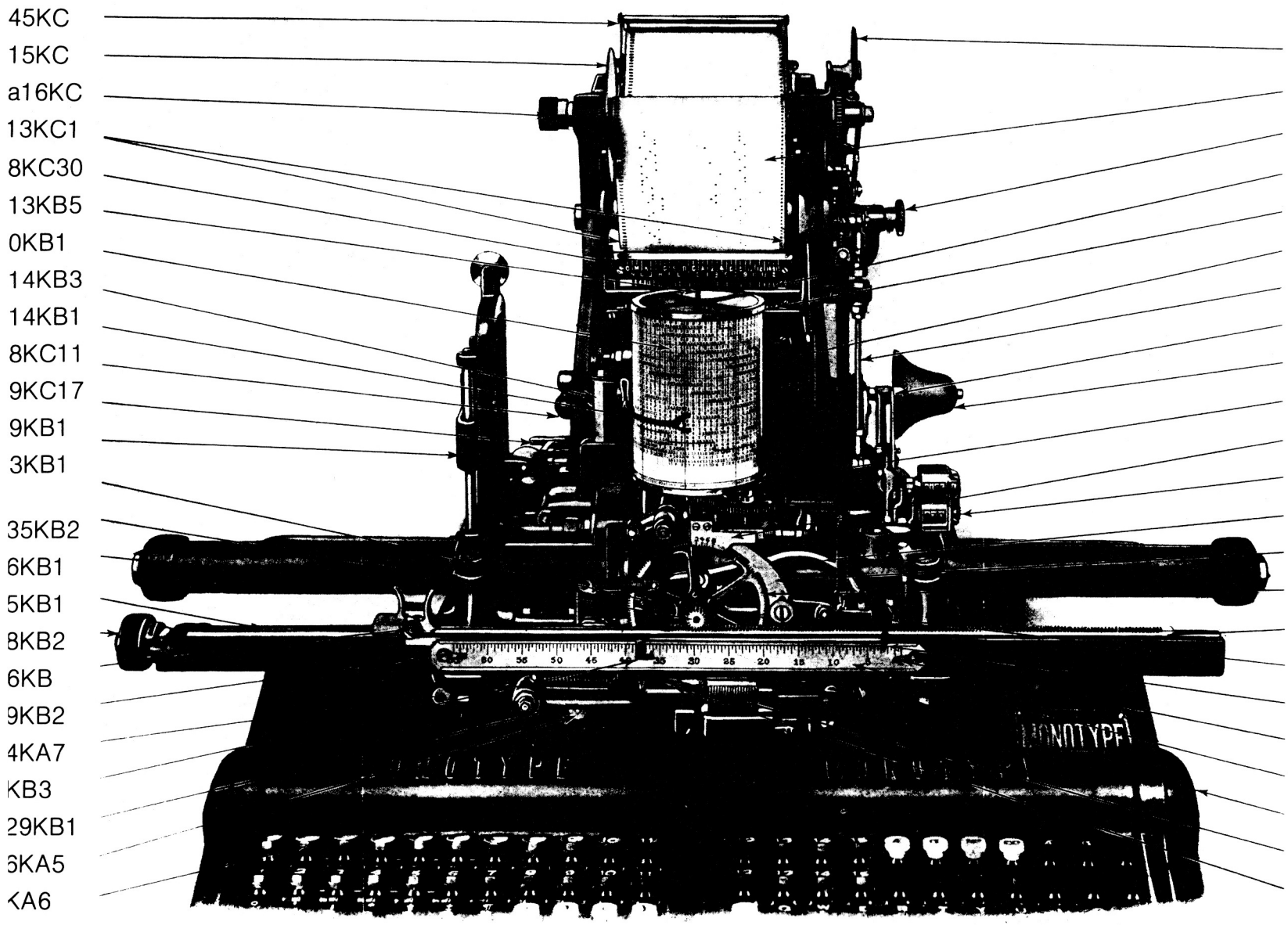
The Em Rack (4KB1) is driven by a Pinion on the Shaft of the Unit Wheel and, therefore, the movement of the Rack is proportional to the movement of the Unit Wheel. Thus, when a seven-unit character is struck, the Wheel revolves seven spaces and the Rack moves to the right seven-eighteenths of an em. This movement of the Rack is measured, on the Em Scale, by the EM Rack Pointer (a4kB3).

The Em Scale (a9KB1) is a strip of celluloid divided into sixty-five ems and each em is subdivided into half-ems. Its chief function is to measure the amount required to complete the line, and therefore, since the Em rack moves to the right as the line progresses, its zero is at the right end.

The number of ems and units required to complete the line or any section of it are shown by the Em Scale and Unit Indicator.

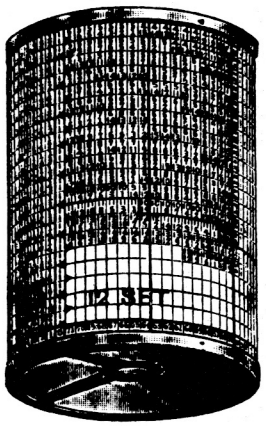
The Justifying Scale (10KB1) indicates the number of justifying spaces in the line, just as the Em Scale shows the number of ems. The lines that run around the surface of the scale divide it into twenty rings and each indicates a justifying space, exactly the same as each graduation on the Em Scale indicates a half-em.

The Justifying-scale Pointer (a14KB1) corresponds to the Em-Rack pointer, for it indicates justifying spaces on the Justifying Scale, by its movement up the scale. The Justifying-scale Pointer is operated by either space bar (right or left keybank) and rises one space on its scale whenever either spacebar is struck, and at the same time the counting mechanism registers four units. After striking the last character in the line, a glance at the Justifying Scale tells the operator the Justifying Keys he must strike to make the spaces between the words the size required to justify the line.


45KC
15KC
a16KC
13KC1
8KC30
13KB5
0KB1
14KB3
14KB1
8KC11
9KC17
9KB1
3KB1
35KB2
6KB1
5KB1
8KB2
6KB
9KB2
4KA7
KB3
29KB1
3KA5
<A6
b8KC2
Holes
13KC10
32KC1
a33KC1
a32KB3
9KC1
2KB1
a1KB1
a41KB1
a25KB1
a23KB1
24KB4
a38KB1
a35KB1
4KB1
3KB1
a4KB3
a9KB5
a9KB1
7KA1
31KB1
c26KB1

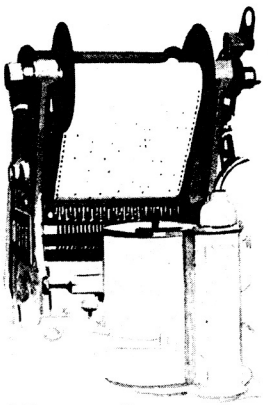
When a character key is depressed, one or two of the punches is forced up by compressed air, the motive power of the original keyboard, through the paper ribbon. The location of these perforations, across the ribbon, determines the characters which the casting machine will cast.

The Casting Machine.—The caster is controlled by these perforations. Its paper feed wheels advance the paper ribbon so that the holes in the tape are positioned (and then clamped) to a location above pipes carrying compressed air. This air can only flow through those pipes not sealed by the juxtaposition of the paper ribbon. The air which so enters regulates the movements of the casting machine, causing it to bring the matrix for the character to be cast over the mold into casting position, and to draw back the mold blade the proper amount to make the type body the width required for the character to be cast.

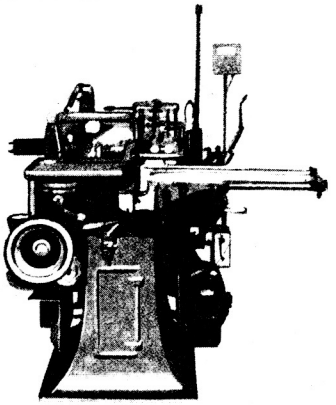


The Justifying Scale

The roll of paper ribbon on the keyboard.

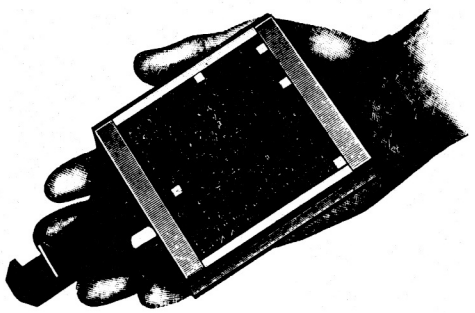


The Monotype Caster

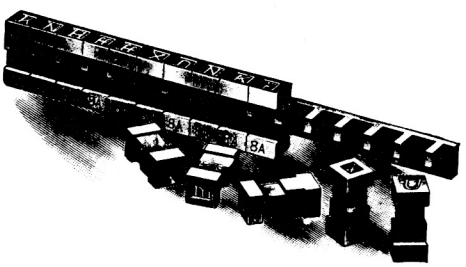


The Matrix and the Matrix Case.—Later versions of the Monotype offer a larger matrix case, but the models most commonly used consisted of a 15 x 15 arrangement of characters—a total of 225 in all, of which 13 are fixed spaces.[1] Individual character matrices can be assembled within the case as desired by the user, but the arrangement must of course correspond to his keyboard layout. The cellular matrices are slipped into the teeth of "combs" which hold the characters firmly, and these combs are then assembled into the case.

1. Subsequently, the mat case was increased. In 1925 a 15 x 17 matrix case was introduced. In 1963 a unit shift feature was developed which virtually made the matrix widths independent of their unit row value. The unit shift arrangement provided for a 16 x 17 matrix case.



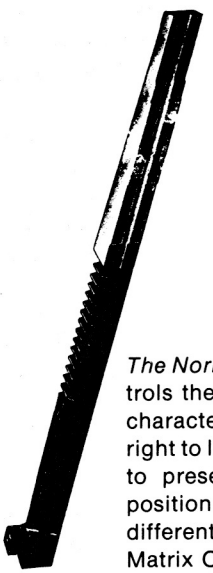
The Matrix Case contains 225 separate Matrices arranged in a square of fifteen on a side. Later models offered a 16 x 17 Matrix Case.



The Matrix Comb: The Matrices are carried between the teeth of the Comb so that each Matrix is practically is own Matrix Case. All Matrices on the same Comb produce characters of the same width (set size).

Characters are arranged within the case so that their widths correspond. In other words, all characters slotted into the same comb must have the same widths. A common, or "standard" arrangement is shown in the diagram below, with one or more rows for each unit size, from five to eighteen, inclusive: three nine unit rows, three ten unit rows—and the 16 and 17 unit sizes are omitted. For offices specializing in tabular work, the keyboard and mat case are often adjusted to provide four nine-unit rows.

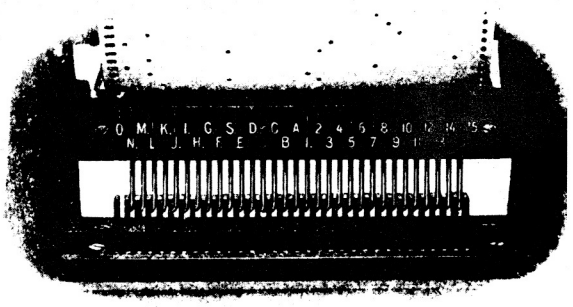
Unit Value	Row	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	Row
5	1	▌	▌	*l*	*t*	‘	’	.	,	▌	l	i	]	[	′	|	1
6	2	*j*	*f*	*i*	!	:	;	-	j	f	I	*l*	:	;	▌	▌	2
7	3	*c*	*r*	*s*	*e*	)	(	’	‘	r	s	t	J	*v*	°	*z*	3
8	4	‡	*q*	*	*b*	*g*	*o*	?	I	z	c	e	z	s	†	*?*	4
9	5	*I*	▌	*9*	*7*	*5*	*3*	*1*	*0*	.	9	7	5	3	1	0	5
9	6	C	▌	▌	*8*	*6*	*4*	*2*	$	–	$	8	6	4	2	▌	6
9	7	*x*	*k*	*y*	*d*	*h*	*a*	x	J	g	o	a	P	F	L	T	7
10	8	A	*fi*	*u*	*n*	.	S	v	y	p	u	n	Q	B	O	E	8
10	9	D	▌	*fl*	*p*	fl	fi	q	k	b	h	d	v	Y	G	R	9
11	10	H	&	*J*	*S*	*œ*	*æ*	*ff*	▌	Z	▌	ff	X	U	K	N	10
12	11	*O*	*L*	*C*	*F*	*w*	£	æ	L	P	F	¶	M	*Z*	*Q*	*G*	11
13	12	*E*	&	Q	V	C	B	T	O	E	A	w	*P*	*T*	*R*	*B*	12
14	13	*D*	*A*	*Y*	*ffl*	*ffi*	*m*	œ	Y	U	G	R	Œ	Æ	w	*V*	13
15	14	*K*	*N*	*H*	ffl	ffi	X	D	N	K	H	m	&	℔	*X*	*U*	14
18	15	Œ	Æ	¾	¼	½	*W*	*M*	—	.	.	M	W	%	Œ	Æ	▌	15
Unit Value	Row	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	Row



The Normal Wedge, which controls the width (set size) of the characters. This moves from right to left with the Matrix Case to present a different wedge position to the Mold for each different numbered row in the Matrix Case.

Two holes punched across a horizontal row of the tape indicate the character to be cast. (In some cases, one hole, or no holes, will suffice.) Punches A to N provide the information that causes the matrix case to move backward and forward (no punch is necessary for row O), and punches 1 through 14 move the mat case left or right. (No punch is required for row 15). Consequently, the space at the lower right-hand corner of the case requires no holes at all in the tape.

The perforations made by the justifying keys which are read at the beginning (*i.e.*, the end) of the line cause the casting machine to position its justifying wedges (before the line is cast) in such a manner that the justifying spaces in that line will be the size required to justify it.



Arrangement of punches: Shows the Index Plate for identifying the 31 punches; viz., fourteen numbers from 1 to 14 inclusive to move the Matrix Case right and left; fourteen indicated by the letters A to N inclusive to move the case front and back; one (.0075) operated by the upper row of Justifying Keys controls the front of the justifying Wedge; one (.0005) operated by the lower row of Justifying Keys for the rear of the justifying Wedge, and one (S) operated by the Space Bars to govern the Space Transfer Wedge. Thus, Punches K and 1 bring the Roman lowercase *i* into casting position.

In addition, there is a "normal wedge" which moves with the mat case and which controls the character's set size. It regulates the distance that the mold blade is pulled back before the type is cast. (A different Normal Wedge is required for each different set size of type.)

Implications of the Monotype system

It is interesting to read literature supplied by the Lanston Monotype Company in the year 1916 because it underscores the virtues of the system at the same time that it seeks to "sell" its capabilities in an environment still (at least in part) dominated by the hand compositor—but nevertheless in a setting in which the linecaster had made substantial inroads:

> Composing-Room Efficiency can be obtained only by keeping clearly in mind this fact: The composing-room of a printing plant is a great deal more than a a place in which to operate composing machines: it is a department maintained to produce complete pages locked-up in chases ready for printing, or electrotyping, or stereotyping.
>
> Now, in the production of complete pages, the work of the hand compositor is quite as essential as the work of the composing machine operator, because no composing machine ever devised can eliminate the hand compositor; his work is essential for making ready for the chases the machine-set matter and also for setting by hand the display matter used with the machine product.
>
> Therefore it is not possible to obtain composing room efficiency if the machinery used in the omposing room fails to provide for the needs of the compositor.
>
> The Monotype is the only composing machine that recognizes the existence of the hand compositor. It is not built upon the impossible theory that the compositor can be driven from the printing industry; instead, it provides the means for increasing his efficiency and making his work highly profitable to his employer.
>
> Not only does the Monotype furnish a product that the compositor can correct and alter without the help of the machine operator, but also, since it is a complete type foundry, it supplies the compositor with all the equipment he requires to work efficiently—type, space material, rules, leads and slugs.

The introduction goes on to claim that the Monotype is the simplest and fastest composing machine—"a machine so free from mechanical limitations that it handles the most intricate composition as easily as straight matter"—and that the system gives much lower cost *for complete pages* "than is possible with any other composing machine," and that the product "is superior in quality to any other process of composition—hand or machine."

The Lanston Company also stressed the advantage of separating the keyboard from the caster:

> The Monotype, two machines in one, is both a type caster and a composing machine; it is the only machine that delivers new type in justified lines in ordinary galleys; it is the only mechanical means for producing printing surfaces far superior to hand-set foundry type, the only composing machine that handles

> straight and intricate matter with equal facility. Its product is corrected and made-up the same as foundry type, it forces no changes in composing-room methods, it requires no special rules, saws or other paraphrenalia; it is built upon the principle that the printer knows best what he needs, and it gives him the type he has always used instead of "something just as good."
>
> Composition is separated from casting by the Monotype System quite as distinctly as in the days of foundry type the type foundry was separated from the composing room; the compositor who uses the Monotype Keyboard gives no more thought to typemaking than does the compositor who sets by hand type bought from a foundry. The Keyboard is as simple, as easy to learn, and as easy to operate as a typewriter, its Key arrangement is the same as any universal typewriter, and no machine has a lighter or more elastic touch. . . The Casting Machine produces every character and space struck by the Keyboard operator when he perforated the ribbon, and the arrangement of these characters and the justification of the lines are exactly as he made them; in short, the ribbon controls the Casting Machine just as the paper roll on a pianola controls a piano. Thus, although the Keyboard operator can direct every movement of the Casting Machine, the two machines are absolutely independent; they may be located together, or apart, as desired, for any ribbon will control any Casting Machine at any place or at any time. A ribbon may be saved and used again for a repeat order, or, for matter that duplicates, as many casts as are desired may be made from the same ribbon.
>
> The advantages of this separation cannot be overestimated. This is an age of specialization and the Monotype applies to the composing room this principle that has made American manufacturing preeminent. The Keyboard operator does not have to be a compositor, a mechanic, and a metallurgist combined. Removed from the fumes and dirt of casting he gives all of his attention to the work for which he is best fitted—composition.
>
> The simplicity of the Keyboard is worthy of special comment: it has been called a "cross between a punch and an adding machine"—certainly it could not have been described more briefly . . . depressing a key does more than perforate the ribbon; it automatically registers the width struck and adds this amount to the sum of the width of the characters previously struck.[1]

1. *The Monotype System*, cited above.

Lanston also stresses the flexibility of the Monotype: "the ease with which it handles the most intricate matter."

But the flexibility of the machine in terms of its character repertoire needs also to be considered. A skilled Monotype operator can arrange both his keyboard and his mat case to offer the specific characters he requires. It is even possible to use "double matrices"—that is, characters twice as high as "normal" characters. The matrix extends beyond the body on which the character is cast. To provide for this overhang, allowance must be made in the line or lines above the one in which the larger characters appear. Quads and spaces equivalent to the width of the larger characters must be set in the appropriate position.

The character repertoire

As we have also seen, the user can arrange whatever selection of characters he desires in his mat case, so long as he achieves a proper allocation of set widths. In other words, he must fill his mat case and if he has three nine-unit rows

he must therefore include in that repertoire 45 characters of that particular width. Most commonly, the 225 characters in the mat case provide for roman, italic, and small caps, as well as ligatures and dipthongs. An alternative arrangement substitutes a bold face for the italic, and it is possible to substitute accented characters for certain of the "signs" which are less frequently used.

On the page which follows we illustrate the manner in which the keyboard layout and the mat case relate to one another, by selecting one of many possible common arrangements.

The concept of "set"

Normally, a Monotype face is designed such that the 18-unit "em" is of the point size specified, but it is possible to design a condensed face in which the "em" of 18 units is wider or narrower than the point size in question. Thus a type face may be specified which is nine points high, to set on an eight-point body width, or, conversely, a face eight points high may be designed to a nine-point set width.

However, it is also possible to require the Monotype system to compose and justify type to a different set width than that for which the type face was designed. By using different-sized "normal wedges" type characters may be cast on wider bodies. Below, we reproduce illustrations of a group of related type faces designed for eight point, eight and one-half set. But these faces can also be "opened up" and are shown in three other versions, each one-quarter set wider than the previous one. It is of course necessary to change the Justifying Scale to correspond with the desired Normal Wedge.

Eight-point faces—eight-and-one-half-set

A NEW IDEA in machinery has been embodied in the latest construction of the MONOTYPE, for, like "elastic" book cases, modern filing cabinets and composing-room furniture, **the Monotype is built up of units** which may be combined to suit the needs of each individual printing office. Thus, the MONOTYPE user can build up his equipment to suit his business exactly, since he can buy just *the units required to fit his individual needs*—**the printer who chooses Monotypes uses "made-to-order" machines.**

Same eight-point faces—eight-and-three-quarters-set

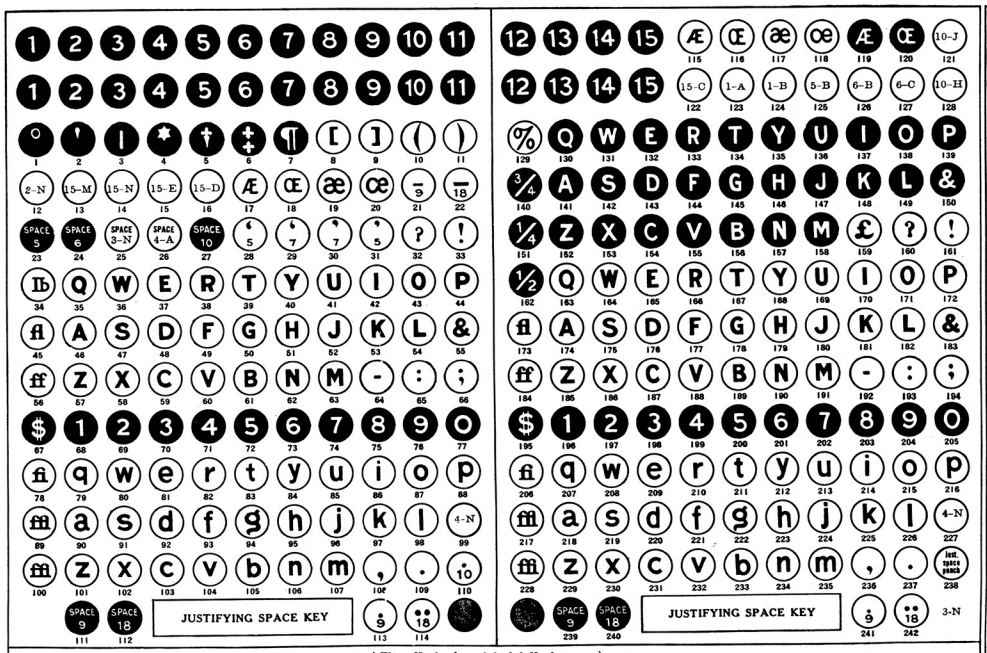
A NEW IDEA in machinery has been embodied in the latest construction of the MONOTYPE, for, like "elastic" book cases, modern filing cabinets and composing-room furniture, **the Monotype is built up of units** which may be combined to suit the needs of each individual printing office. Thus, the MONOTYPE user can build up his equipment to suit his business exactly, since he can buy just *the units required to fit his individual needs*—**the printer who chooses Monotypes uses "made-to-order" machines.**

Same eight-point faces—nine-set

A NEW IDEA in machinery has been embodied in the latest construction of the MONOTYPE, for, like "elastic" book cases, modern filing cabinets and composing-room furniture, **the Monotype is built up of units** which may be combined to suit the needs of each individual printing office. Thus, the MONOTYPE user can build up his equipment to suit his business exactly, since he can buy just *the units required to fit his individual needs*—**the printer who chooses Monotypes uses "made-to-order" machines.**

Same eight-point faces—nine-and-one-quarter-set

A NEW IDEA in machinery has been embodied in the latest construction of the MONOTYPE, for, like "elastic" book cases, modern filing cabinets and composing-room furniture, **the Monotype is built up of units** which may be combined to suit the needs of each individual printing office. Thus, the MONOTYPE user can build up his equipment to suit his business exactly, since he can buy just *the units required to fit his individual needs*—**the printer who chooses Monotypes uses "made-to-order" machines.**



Left Keybank C—use Keybars C { These Keybanks and the left Keybars are used also with Arrangements C1 and C2 (Job). See their Charts for cappings. } **Right Keybank C—use Keybars C**

The extra Keys not used with this Matrix Case Arrangement (shown at right) are indicated by figures and letters printed in red within the circles. These red figures and letters show the Matrix Case positions produced by these extra Keys, which are used on special work for additional characters not carried in the Matrix Case.

Row	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	
1	123	124	226	211	28	31	109-237	108-236	23	98	86	9	8	2	3	5 Unit
2	224	221	214	33	65	66	64-192	96	93	137	161	193	194	12	1 unit if justifying space. See Note.	6 Unit
3	231	210	219	209	11	10	30	29	82	91	83	147	232	1-25-243	229	7 Unit
4	6-26	207	4	233	222	215	32	42	101	103	81	152	142	5-227-99	160	8 Unit
5	170	125	204	202	200	198	196	205	113-241	76	74	72	70	68	77	9 Unit
6	154	126	127	203	201	199	197	195	21	67	75	73	71	69	111-239	9 Unit
7	225	212	220	223	218	102	52	94	87	90	139	144	149	134		9 Unit
8	141	206	213	234	110	47	104	84	88	85	106	130	156	138	132	10 Unit
9	143	27	173	216	45	78	79	97	105	95	92	155	135	145	133	10 Unit
10	146	150	180	175	118	117	184	128	57	121	56	153	136	148	157	11 Unit
11	171	182	187	177	208	159	19	54	44	49	7	158	185	163	178	12 Unit
12	165	55	35	60	59	61	39	43	37	46	80	172	167	166	189	13 Unit
13	176	174	168	217	228	235	20	40	41	50	38	120	119	131	188	14 Unit
14	181	190	179	89	100	58	48	62	53	51	107	183	34	186	169	15 Unit
15	18	17	140-122	151-16	162-15	164	191	22	114-242	63	36	129	116-13	115-14	112-240	18 Unit

Characters in Matrix Case

ABCDEFGHIJKLMNOPQRSTUVWXYZ&ÆŒ

ABCDEFGHIJKLMNOPQRSTUVWXYZ&ÆŒ

abcdefghijklmnopqrstuvwxyzæœfiflffifffifffl

ABCDEFGHIJKLMNOPQRSTUVWXYZ&ÆŒ

abcdefghijklmnopqrstuvwxyzæœfiflffifffifffl

.,-;:''!?()[]°*†‡¶—–...|%£lb;:!?''.

$1234567890½¼¾$1234567890

Note: Both the Justifying-space Bars and Key 24 produce Matrix Case position 2-O. The Space Bars also bring up the Justifying-space Punch and consequently produce Justifying spaces. Key 24 produces 6 unit spaces because it does not bring up the Justifying-space Punch.

The red numbers in the squares with the characters indicate the Keys that produce these characters.

See Keyboard Arrangement at left.

¼ 151-16 Two or more numbers in the same square with a character indicate the different Keys that will produce this character; for example ¼ will be produced by either Key 151 or Key 16.

On special work these extra Keys are used for additional characters: Thus, Key 16 might be capped with any desired character of same width body as ¼, for which this extra character would be substituted, without affecting the justification, when the matter is corrected.

Keyboard and Matrix Case Arrangement C (Book)—PLATE IX

Effect on type design

The Monotype System—although it required relating all type characters to a common base of 18—did not in fact impose unreasonable restraints upon type designers, and the ability to position characters in whatever locations were necessary (so that they could take on the width values assigned to that row) made it possible to exercise a great deal of freedom in the design of individual characters. Unlike the linecaster, which commonly "duplexed" bold and roman, or roman and italic, these faces could take on widths more appropriate to their characteristics.

But to deviate from a "standard" layout meant "reprogramming" (if that word can be so used) the keyboard. To do this you had to change keybanks and keybars and mount the proper Justifying Scale. If the only change you wanted to make was to set a different point size you had only to replace the Justifying Scale. But a different mat case layout would require replacement of keybars and perhaps also the keybanks as well. Despite the contention that the keyboard is as easy to use as a typewriter, it is evident that the price one pays for "flexibility" in character repertoire is that the operator who moves from one layout to another must concentrate to remember that certain key positions have been reassigned. It is also clear that a change from one layout to another is not a trivial undertaking, although it is easily accomplished by experts. Remember that there is no "hard copy"—that is, no representation of the characters imprinted upon a sheet of paper—and no "soft copy" either—that is, some sort of visual display, when struck, of the desired character. All that you see is a hole punched in the tape, which means only that a certain position on the matrix case has been addressed.

What Monotype teaches us

One thing we must conclude from the Monotype experience is that typesetting devices must service a wide variety of needs. The Monotype solution, developed at a time when the link between cause and effect was achieved by means which were primarily mechanical rather than electronic, illustrates that human ingenuity has not been substantially limited by the necessity of devising solutions so mechanically complex. On the other hand, today's methods are not nearly so cumbersome, and are often so unobtrusive (*i.e.*, "transparent") that we may not appreciate the complexities involved, and the intricacies of the electronic solutions. Someone has had to think all of these same problems through, taking into consideration every conceivable case, in order to arrive at a "general purpose" answer.

Review Questions:

1. Describe the functions performed by the Monotype's counting mechanism.
2. How does the paper ribbon generated at the keyboard drive the caster?
3. How are the type matrices arranged in the matrix case? Are characters always located in the same positions? If not, why not?
4. What is the function of the "Normal Wedge"?
5. Describe the justification process, including functions of both keyboard and casting machine.
6. Monotype claims that their machines possess greater flexibility in operation and composition than linecasters. How would they substantiate that claim?
7. How extensive is the character repertoire?
8. What effects did the Monotype system have on type design?
9. Is it necessary to "duplex" italics to match roman on the Monotype?
10. What advantages do you see in having variable set size?
11. What meanings of "set" are relevant in discussing Monotype composition?
12. Why does the Monotype paper ribbon require so many punch positions and contain so few punch holes per row? (Compare this with the TTS coding structure described in the next chapter.)
13. Why is it sometimes necessary to replace one keybank with another when a different matrix is used?
14. How would you expect corrections to be made of Monotype galleys, as opposed to the method of correcting Linotype galleys?

6

The Advent of TTS

IN THE EARLY 1930s a new method of setting type on linecasters was introduced. This is a method known as *TTS*. The initials stand for "teletype-setting" because it was a system developed to send impulses over a telegraph or telephone line, thus permitting type to be set at remote locations. The inventor of the teletypesetter system was Walter Morey, whose background included work as a composing room superintendent and as a Monotype salesman. He began his experiments in the 1920s in Newark, N.J., and later received financial backing from Frank Gannett, a newspaper publisher in Rochester, N.Y., who founded the Gannett newspaper chain.

Morey's invention came to fruition in the early 1930s, but was hardly used until after World War II. The most practical application for TTS was what we call the "wire services." These are news-gathering agencies such as UPI (United Press International) and AP (Associated Press). At that time reporters would phone or otherwise relay their stories to a central news bureau. Here the stories were edited, channelled for distribution according to interest and category of news, and transmitted over the "teletype" to local newspapers. There they were received on teletype "printers." Local editors could read and garner information from these reports, rewriting what was relevant to them, or send them on to the composing room to be typeset, as received, in their entirety.

Morey visualized the possibility that stories could be transmitted in "justified" form and could be received not only as "printout" but also as punched paper tape capable of activating a linecaster. Tape from stories selected for publication would thus be available for setting *without the necessity of re-keyboarding.*

The wire services were slow to begin the transmission of pre-justified stories, keyed and justified at the transmitting end. There were many obstacles in the way, such as the absence of a uniform typographical format on the part of the newspapers, their uncertainty about whether or not they wished to use the wire service stories more or less intact, and union objections to the elimination

of local keyboarding. Nevertheless, in 1951 both AP and UPI did begin the transmission of such material. They were encouraged to do so in part because a certain number of newspapers, particularly in the south, (notably in North Carolina and Arkansas), were using TTS input internally, and were therefore equipped to receive it.

The "internal" use of TTS

The fact is, therefore, that from 1932 until the 1950s, TTS hardly caught on. and when it did it was primarily employed not as a consequence of teletype transmission but for whatever advantages it offered for purely local production. Perhaps the major exception was in the case of Time, Inc., where, for some years, beginning in the 1940s, material keyboarded in Time's editorial offices in New York was transmitted to several locations (notably Chicago and Philadelphia) where the signals were recorded (punched) onto paper tape which, in turn, was mounted on linecasters. But by the late 1950s this practice of multiple transmission and setting was discontinued as air travel became quicker and more reliable. Electrotype shells were flown from Chicago to Philadelphia and elsewhere. But TTS transmission still took place between the editorial offices in New York and the plant of the printer (R. R. Donnelley) in Chicago.

TTS is a method more suited to straight text than for the setting of more complex work, and, at least initially, it could only be used in situations in which typesetting with unit-count fonts was acceptable.

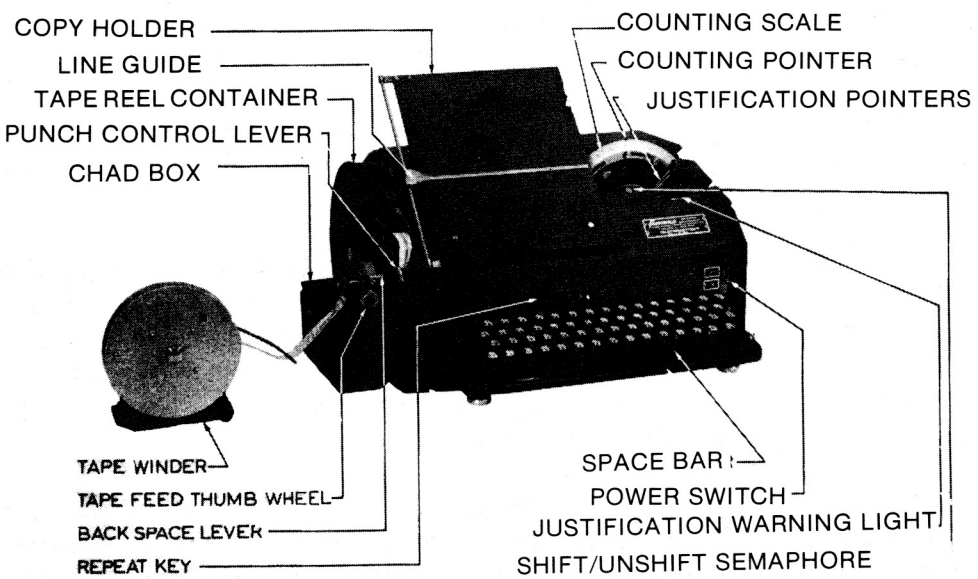
As we have already explained,[1] linecasting fonts were not designed with any unit counting system in mind. The operator made his own end-of-line decisions on the basis of his visual judgment. But if the keyboard is to be separated from the "caster," so that the keyboarder can no longer see the matrices fall into the assembler, then some method of counting is essential so that he can determine where to place his line-ending decisions.

1. *See* page 5.

To use TTS, initially, it was necessary to develop a method of counting at the keyboard, and this led to the creation of so-called "unit-cut" or "unit-count" fonts. Modifications of existing art work of type masters were drawn which—as with Monotype—made use of the 18 unit system. Unlike Monotype, where four different faces can be varied according to the manner in which characters are positioned in the mat case, there was initially no such flexibility with respect to type design for the linecaster. And so, regardless of type face or style, each character had to be redesigned to take on an appearance that could be described by an assigned universal width. A Times Roman lower case *a* had to have the same unit value as a Spartan, Trade Gothic or Corona *a* of the same point size. Moreover, the character repertoire itself was more or less standardized. Hence a "standard" perforator (tape-perforating keyboard) could be employed. Every newspaper receiving the wire service copy would set it in an 11-pica measure, in nine point type of whatever face was chosen, so long as its width values were those of a "unit-count font." Later, as we shall see—as with Time, Inc.—other ("unitized" and non-uniform) type faces could be chosen.

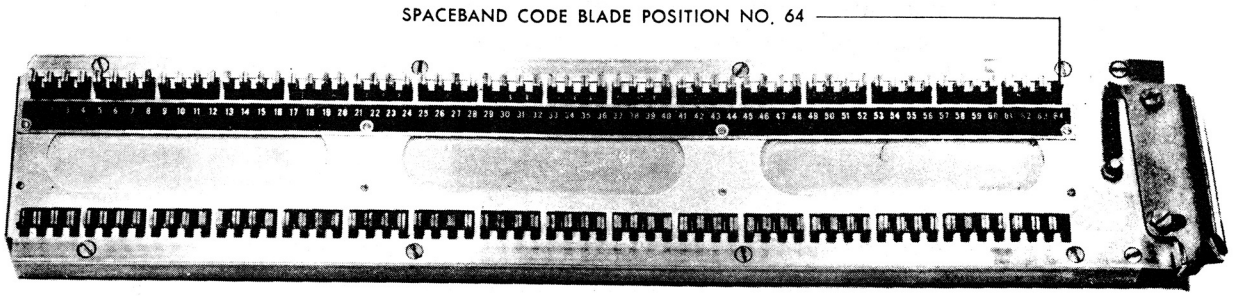
The TTS Keyboard

Like the Monotype keyboard, the original TTS Standard Perforator performed two basic functions—it punched coded holes in paper tape and it provided a counting mechanism. Pressure on the keys activated selector bars. These made electrical contacts which sent signals to the solenoids in the punch mechanism. The same selector bars provided information to a "squirrel cage" which rotated according to the unit count of the character. This rotation caused a pointer or scale to move. (In earlier models, affectionately known to the trade as the Fairchild *brownies*, the scale moved and the pointers remained constant. On the later models —the *greenies*—the scale was fixed and the pointers moved.) One pointer, on the left of the scale, indicated the increasing accumulation of character widths, while two other pointers, on the right, indicated the amount of space taken up by the spacebands as calculated, and their minimum and maximum values. When the left pointer found itself within the area which measured the difference between minimum and maximum interword space expansion, the operator knew he was "within justification range" and could "release the line" by striking the "return" and "elevate" keys.

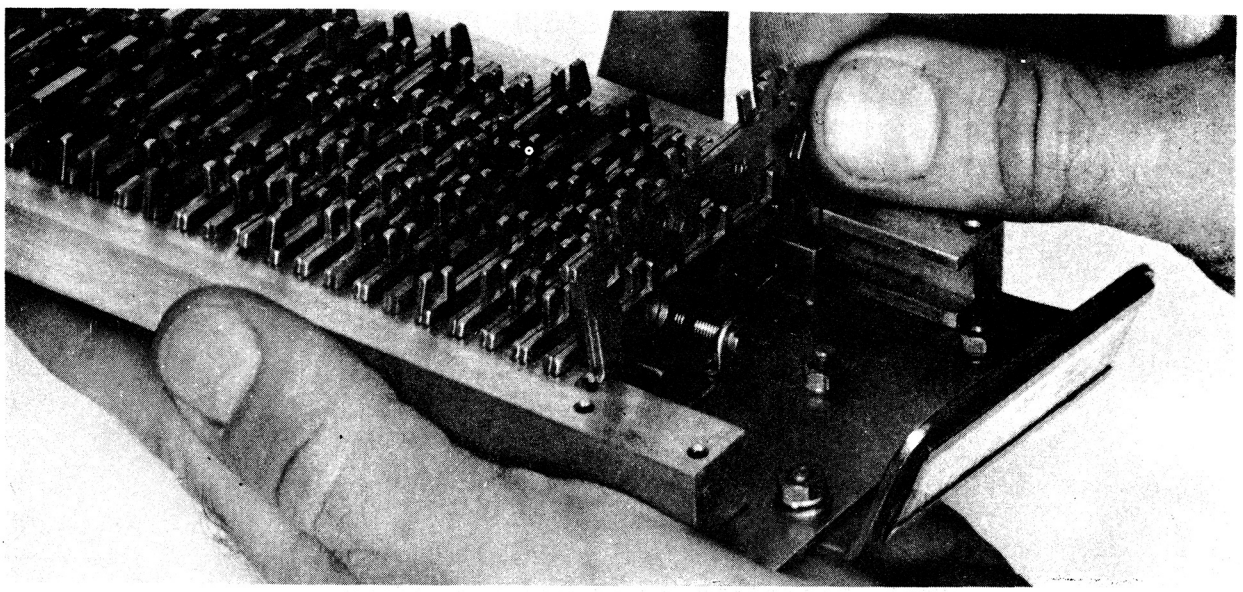


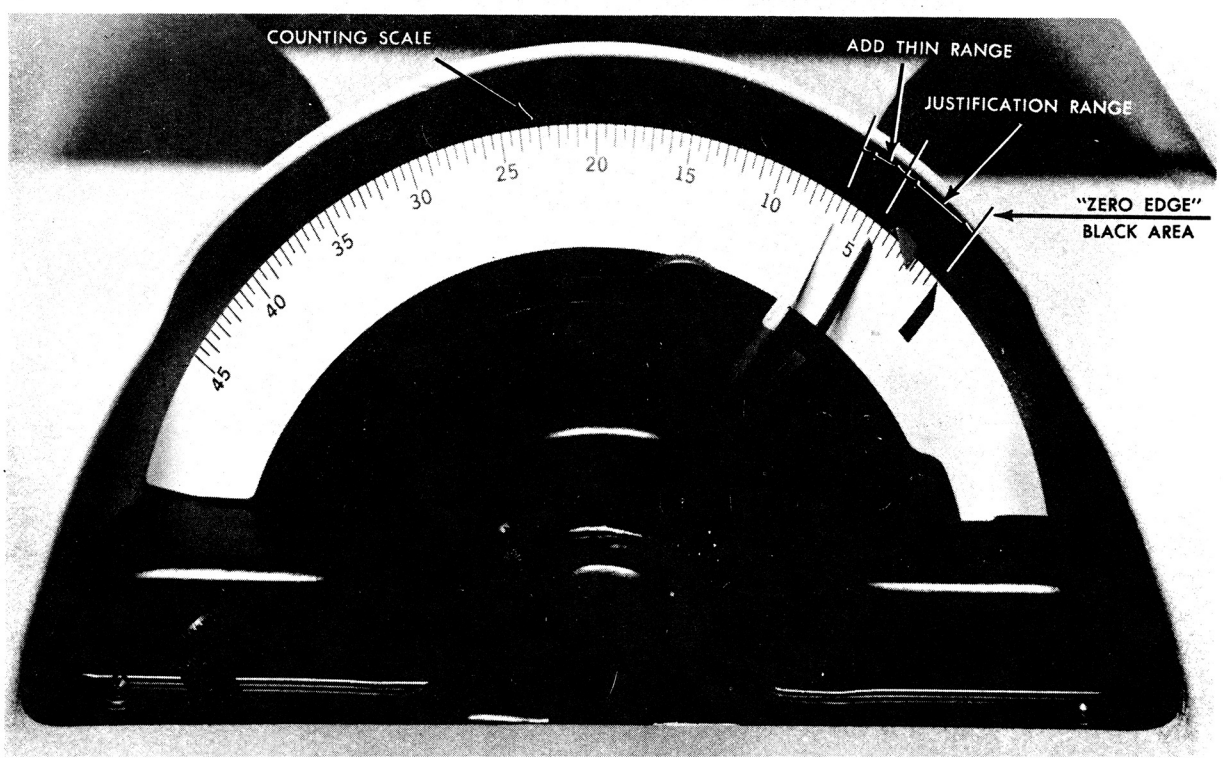
Subsequently, the "multiface perforator" was introduced. This device included a "counting magazine" consisting of a series of blades which were contacted by the selector bars, and which in turn activated the squirrel cage. The movement of the squirrel cage, and hence its pointers, was determined by the location of the blades in the counting magazine, which designated character widths.

The multiface perforator was based upon a 32 unit-to-the-em system. A separate counting "magazine" had to be inserted for each change of type face. The "brass width" values of the linecaster matrices were rounded off to the nearest 1/32of an em. For example, a nine-point em should measure .1243",



TELETYPESETTER COUNTING MAGAZINE

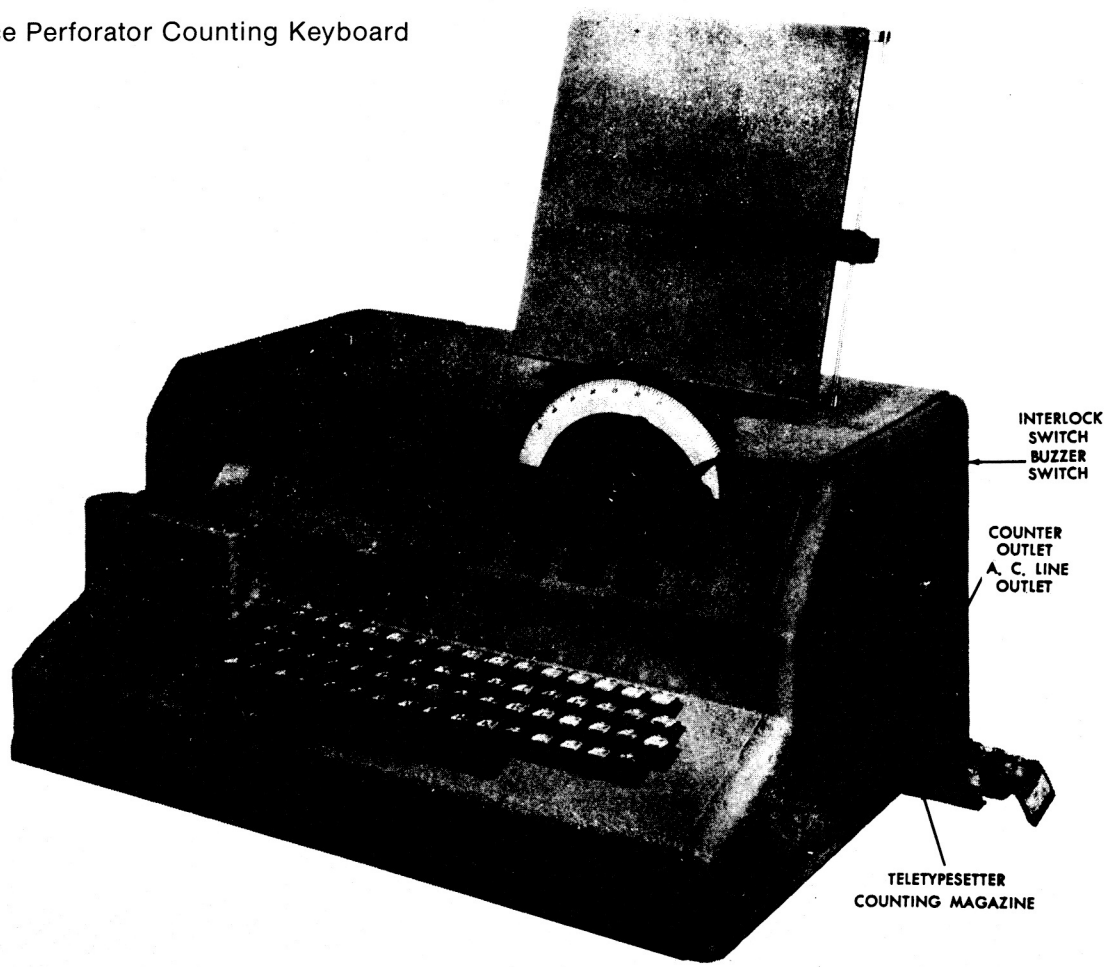




and thus 1/32 of a nine-point em would equal .0038812″. In order to assign "unit values" to type not so designed, the brass width of each character would have to be measured by a micrometer. If a lower case *a* "miked" at .063″, and you divided this figure by .0038812 you would derive 16.23 units, which would then be rounded off to 16 relative units. These rounding errors, being plus or minus .00194″ (in our illustration) would tend to cancel each other out. But if they did not do so—giving consideration to the frequent use of certain characters (which might be "over"— as opposed to others (which might be "under")—you could be "over" or "under" by as much as .05″ in an 11 pica line. This might—in relatively rare instances—cause a line to be set either too tight or too loose to cast properly.

Nevertheless, so far as we know, hot metal type was never redesigned for multiface TTS operation on the basis of the 32-unit system, although type was designed for unit-count applications. As we have indicated, in the 1950s the wire services did begin to transmit pre-justified tapes—and the transmission of stock market data also occurred. More and more newspapers therefore converted to unit-count designs so that they could avoid rekeyboarding.

The Multiface Perforator Counting Keyboard



On the other hand, some book and commercial shops made fairly extensive use of multiface perforators. The principal advantage appears to have been the higher productivity of the TTS keyboard operator as compared with the linecaster operator at his machine. Even after allowance is made for the fact that a "monitor" would be required for every two, three, or perhaps four tape-driven linecasters, total keystrokes produced, divided by total man hours, seem in general to have increased. And in some cases acute labor shortages were alleviated, since TTS keyboards could be operated by persons with typing skills (plus some additional training), who did not need to serve a formal four- or six-year apprenticeship. And by adding more keyboards on the "day side," it was possible to keep linecasting machines busy on the "night end" and even on the "lobster shift" without the need to pay night differentials, and without the problem of recruiting additional personnel for night work.

Let us turn our attention to the manner in which the linecaster is driven by TTS tape. Characteristically, this tape is (or was) "read" by the "fingers" of a mechanical paper tape reader, which sensed the presence or absence of holes in the paper tape and conveyed electrical signals to an adaptor keyboard, under which an "operating unit" depressed the keys that released the matrices. The same operating unit also controlled the "rail" in the assembling elevator (to access the alternate duplexed font of italic or bold) and caused the elevator to rise in order to cast the line and to redistribute the matrices. Subsequently, the operating unit was enhanced to be able to handle the functions of an "automatic quadder" which filled out short lines at the end of paragraphs with the necessary fixed spaces.

Later models of linecasters, beginning about 1959, (such as the Intertype Monarch and the Linotype Elektron) were built to read paper tape without the need for an adaptor keyboard or operating unit. Such machines were capable of casting from ten to fifteen newspaper lines per minute, although the latter figure was seldom achieved. Twelve lines per minute is the equivalent of approximately six characters per second (whereas a Monotype caster could not quite achieve a speed of three characters per second.)

The paper tape coding

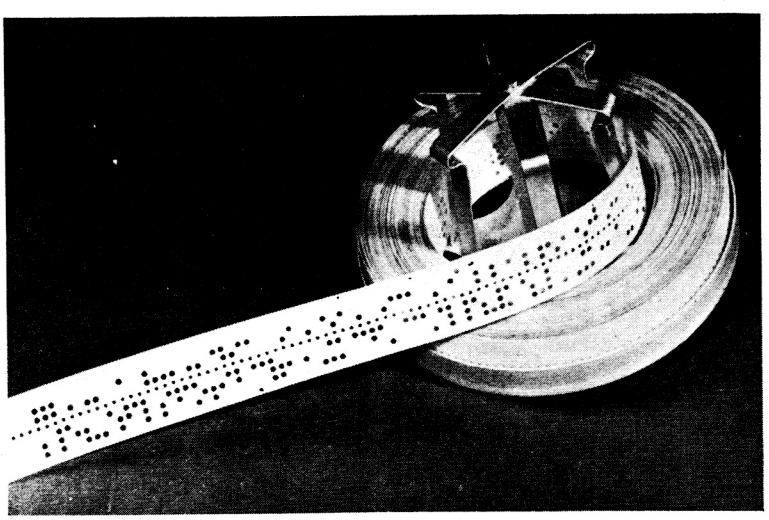
The six-level code which drives the linecaster was developed by Walter Morey, inventor of the TTS system, as an adaptation of the five-level code which had been used in message transmission over telegraph and telephone lines. As the reader doubtless knows, telegraphers originally used the Morse code, consisting of dots and dashes (*circa 1840*). Several subsequent inventions sought to devise a method of semi-automatic transmission, one of which was the "Wheatstone Automatic Telegraph." Professor Charles Wheatstone developed a centered sprocket tape which recorded a dot by punching a hole on one side of the sprocket feed, and a dash by punching a hole first on one side and then (in the next position) on the opposite side.

Because each letter consisted of different combinations of dots and dashes—some more lengthy than others—a blank sprocket position separated the characters.

Subsequently, Emile Baudot (1845–1903), a French engineer who pioneered long-distance telegraphic installations (as from Paris to Vienna), developed a five-level code which was produced by a "chord" keyboard consisting of five keys. The signal from combinations of these five keys was picked up serially—that is, each of the five components (present or absent) was transmitted in sequence, preceded and followed by a start and stop signal. (The normal operating speed of this method was considered as 30 words per minute, and this led to our use of the term "baud" to describe serial transmission speeds.)

A. C. Booth developed a more conventional typewriter-like keyboard which also produced the Baudot code, but now as a *frame* across a paper tape, and a subsequent variation (1901) called the *Murray* code (known in the U.S. as the Baudot code) made use of an advanced sprocket hole to indicate the direction of the paper tape.

The "Murray" code affords a combination of 32 possibilities (assuming "no holes" to be one such). The 26 alphabetic characters and a space were supplemented by a carriage return and a "line feed." The other two codes were used to signal a "shift" from "letters" to "figures" and back to "letters" again. The "figure" mode included some punctuation marks as well as the ten digits. Obviously there were no lower-case alphabetic characters.

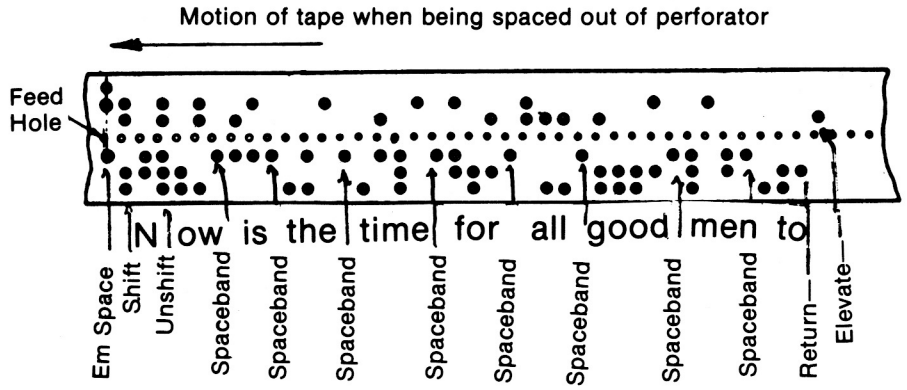


The basic principle behind the assignment of "holes" across the tape was that the wear on the tape punch should be minimized. Hence what were taken to be the most frequently-used characters were represented by the fewest holes. The letters "e" and "t," as well as "space" required only one hole each, and *s*, *h*, *r*, *d*, and *l* needed but two. It may be readily seen that Morey created a whole new series of codes, since the addition of the sixth level (actually called "row 0") added 32 new possibilities. These were used for the most part to meet typesetting requirements—quad left, right and center, em, en, thin and add thin spaces, upper rail and lower rail. Also the "letters" and "figures" concept gave way to the upper case and lower case. The figures and special characters were thus assigned to "sixth level" ("0" row) positions in a shift or unshift mode.

Morey's adaptation of the Murray or "Baudot" coding is illustrated on the next page. On page 66 a diagram of a paper tape reader and tape configuration is

Table I—The Morey (TTS) Adaptation of Murray-Baudot Coding

Murray			*TTS*		
	Letters Mode	Figures Mode		Downshift Mode	Upshift Mode
11 000	A	hyphen	011 000	a	A
10 011	B	?	010 011	b	B
01 110	C	:	001 110	c	C
10 010	D	$	010 010	d	D
10 000	E	3	010 000	e	E
10 110	F	! or *	010 110	f	F
01 011	G	& or *	001 011	g	G
00 101	H	★	001 101	h	H
01 100	I	8	001 100	i	I
11 010	J	Bell	011 010	j	J
11 110	K	)	011 110	k	K
01 001	L	(	001 001	l	L
00 111	M	.	000 111	m	M
00 110	N	,	000 111	n	N
00 011	O	9	000 011	o	O
01 101	P	0	001 011	p	P
11 101	Q	1	011 101	q	Q
01 010	R	4	001 010	r	R
10 100	S	apos.	010 100	s	S
00 001	T	5	000 001	t	T
11 100	U	7	011 100	u	U
01 111	V	=	001 111	v	V
11 001	W	2	011 001	w	W
10 111	X	/	010 111	x	X
10 101	Y	6	010 101	y	Y
10 001	Z	+	010 001	z	Z
00 010	carriage	return	000 010	carriage	return
01 000	line	feed	001 000	line	feed
11 111	Letters	Letters	011 111	Unshift	Unshift
11 011	Figures	Figures	011 011	Shift	Shift
00 100	Space	Space	000 100	Space	Space
			100 000	Thin space	Thin space
			110 000	3	?
			101 000	lower	magazine
			111 000	|	!
			100 100	Add thin	Add thin
			110 100	Em space	Em space
			101 100	8	—
			111 100	7	&
			100 010	'	'
			110 010	-	@
			101 010	4	*
			111 010	Bell	Bell
			100 110	,	,
			110 110	Quad left	Quad left
			101 110	En space	En space
			101 101	0	?
			111 101	En ldr.	En ldr.
			100 011	9	option
			110 011	Upper rail	Upper rail
			101 011	;	:
			111 011	Lower rail	Lower rail
			100 111	.	.
			110 111	1	⅛
			101 111	Quad cent.	Quad cent.
			111 111	Rub out	Rub out



Above: Six-level paper tape carrying TTS codes. *To the right:* A paper tape punch. *Below:* Table II illustrates how the Murray codes (and the TTS codes derived therefrom) attempt to minimize wear and tear on the punch by selecting the fewest holes for the most common characters. Note that linecaster channel assignments are comparable.

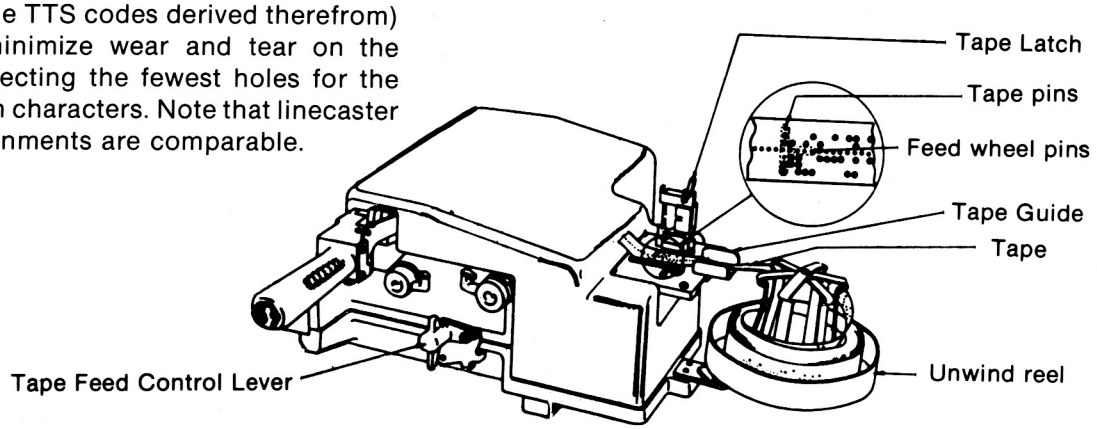


Table II

Linecaster Channel No.	*No. of holes in Murray code*	*No. of holes in TTS code*	*Character*	*Frequency*
--	1	1	spaceband	7603
1	1	1	E	3911
2	1	1	T	2769
3	1	1	A	2327
4	2	2	O	2631
5	2	2	I	1821
6	2	2	N	2087
7	2	2	S	1849
8	2	2	H	2134
9	2	2	R	1739
10	2	2	D	1678
11	2	2	L	1422
12	3	3	U	922

shown, followed by Table II, which indicates how the codes assigned correspond to the frequency of the use of characters as indicated by their positions in the channels of a linecaster.[1]

1. *See Phillips, Arthur H., Computer Peripherals and Typesetting*, London, 1968, H.M.S.O., esp. pages 425-6.

Analysis of TTS code assignments

Out of the 64 available codes, there are some that are 'universal" in the sense that they do not take on a different meaning according to "mode." They have the same significance in the "shift" and "unshift" positions, and in the "upper-" and "lower-rail" positions, as well as in the basic and alternate magazine assignments.

These codes are the following:

Tape feed (no holes)
Carriage return
Space band
Elevate
Shift
Unshift
Thin space
Lower magazine
Add thin space
Em space
Hyphen
Bell
Comma
Quad left
En space
Upper magazine
Em leader
En leader
Upper rail
Lower rail
Period
Quad center
Rub out (all holes)

In a few cases, these codes actually stand for characters—and characters which, like the weight and shape of a period, comma or hyphen, are font-related. But for the most part these codes are concerned with machine operating instructions. When we subtract them from the available 64 we find that 39 or 40 (a few more if you count the period, comma and hyphen) are "mode-related." These are the 26 "alpha" characters and the ten numbers, plus those codes which, in their shift, and unshift modes, perform more or less parallel functions: the left and right quotes, the left and right parentheses, and the colon/semicolon. However, there are ways of using even these few codes to extend the character repertoire. This opportunity presents itself primarily in the upper-shift numbers positions. So, for example, you may have the following arrangements:[2]

2. The letters shown in the table below, which look like capitals, are "small caps." The combination characters, like "ae" or "fl" are either diphthongs or ligatures.

Lower Rail		Upper Rail	
Downshift	*Upshift*	*Downshift*	*Upshift*
0	fl	J	Y
1	ffi	V	W
3	?	C	?
4	*	D	O
5	ff	E	L
6	ae	F	K
7	&	G	N
8	—	R	—
9	fi	T	option
0	fl	J	Y
|	!	P	option
;	:	;	S
)	(	Q	A
-	oe	-	H

The use of a precedence code or mode determiner

An examination of the code conventions shown above illustrates the manner in which one set of codes can do "double duty." With the Murray-Baudot five-level code, for example, there are two "modes" which are represented by the same combination of "holes" and "no holes" on paper tape. The person doing the keyboarding will indicate by a *precedence code* or *flag* which mode he intends. This means that of the 32 available code combinations in the five-level code structure, one code must be reserved to signal the *entering* of one mode and another must be reserved to signal *departure* from that mode and hence entrance into the alternate mode. One mode represents the alphabet, and the second is designated for "figures" and other miscellaneous characters. If the first mode is not signalled, perhaps it may be assumed, "by default," that the mode will govern.

The precedence code injects into the text stream an "until-further-notice" condition. In other words, the new mode governs until the decoder (the teleprinter, for example, that receives the signals) is "told" otherwise. This makes it important, in "processing" or "reading" the paper tape, to observe the sequential nature of the input stream, for if you moved about at random, using only a portion of the incoming tape, rather than starting from its beginning, the decoder would not "know" what mode would govern.

Thus, in a sense, the precedence code becomes a "switch" and when it is encountered an alternate set of symbols is substituted. To use computer terms—which will become more familiar to us later—we might say that the precedence code throws a "switch" (which causes a program "branch") or which acts otherwise to direct the use of a different set of *translate tables*—that is, the use of some method of substituting one set of symbols for another. We might also say that we substituted one *overlay* of characters for another, within the same code structure.

The six-level TTS code offers 64 combinations of holes and no-holes if we count "all no-holes" as one of the combinations. In this case, however, we encounter two different kinds of precedence codes. One differentiates upper case characters from lower case characters. You will observe that the code combination of holes for a lower-case *a* and an upper-case *A* are precisely the same. The decoder can only know which of the two symbols is intended by the use of a precedence code which states that "until further notice I intend you to use only lower-case characters." This is precisely what happens on a typewriter when you press the shift lock key. YOU WILL REMAIN IN CAPITAL LETTERS UNTIL YOU Turn off the condition. When you unlock the shiftlock you are creating (figuratively) an "unshift" or "downshift" code.

On a linecasting machine, however, unlike the typewriter, the upper-case characters are not on the same matrix as the lower-case characters. On a typewriter with striking bars, the lower-case *a* and the upper-case *A* are not only accessed by the same key, but are created by the same keybar. A movement of the

carriage enables the alternate character to impact the ribbon. (This is something like a "rail" change on a linecaster, but here it is a "shift" mode change.) On the Selectric "golf ball" typewriter, the same key, when struck in a shift or unshift mode, must access a *different position on the typing element.* (Upper-case characters are 180° removed from lower-case ones.)

Getting back to the linecasting machine, the reason why the lower-case *a* is not on the same matrix as the upper-case *A* is that the brass width of the upper-case character is greater than for the lower-case character, and only characters with matching widths may be engraved upon the same matrix.[1]

1. For this reason, separate linecaster keys activate upper case characters, and the "shift" conventions of the typewriter do not govern.

Note that the "upper-case" condition is more confuisng when we move away from the "alpha" characters. On a typewriter, the "shift" symbols above the numbers are more or less arbitrarily assigned, although there now tends to be some degree of standardization. There is less standardization with respect to the use of the available upshift positions (above numbers) when it comes to linecasting. The assignment of these codes often depends upon the nature of the character set required—whether for "book" or "job" work, for example, since in the former case SMALL CAPS and ligatures may be desired, and other possible symbols may have to be sacrificed—such as the one piece fractions (⅛, ¼, ⅜, ½, ⅝, ⅞, *etc.*)

TTS provides another set of precedence codes besides those required to indicate the difference between upper and lower case. This has to do with a shift from "lower-" to "upper-rail."

In the lower-rail position you will generally set in a "roman" face. In the upper-rail position you will generally set either an italic or a bold face. When you provide the TTS encoder with a flag that sends the machine into upper rail, it will remain in upper rail until further notice, and hence the linecaster positions the matrices on the rail so that they will cast the alternate character engraved on the matrix. The alternate type face is "duplexed" so that a lower-case roman "a" will correspond in width with a lower-case italic (or bold) "a." In this instance, then, the code does in fact cause the linecaster to access the same matrix. The A and the *A* are paired together, and the a and the *a* are also duplexed and paired (or, alternately, a and **a**).

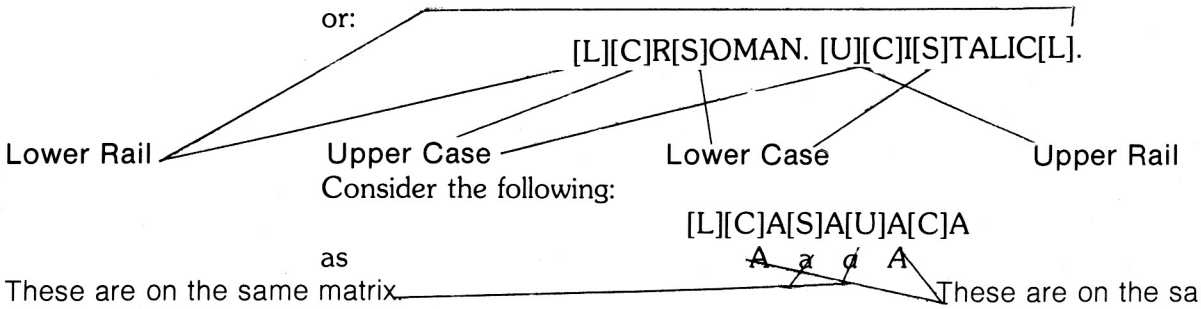
Now we can see that of the 64 available codes, two have been pre-empted to indicate "shift" and "unshift," and two to indicate "upper" and "lower" rail. Bear in mind, then, that these codes open up parallel tracks. In other words, those codes which are generally used to describe characters (as opposed to functions) now have four alternate meanings, depending upon their mode. The same code can mean:

a
a
A
A

Or, alternately, a, *a*, A, and *a*.

As an exercise, let us suppose that we represent each code as a capital letter in order to convey the impression that the code itself is "neutral" with respect to modes. And let us represent an "upshift" as [C] for "capital" and a downshift as [S] for "small letter," bearing in mind that the [C] and [S] codes are really only one character and one keystroke each. In like manner, let us represent the basic face as [L] for "lower rail" and [U] for "upper rail."

[L][C]T[S]HIS IS THE WAY THE TEXT STREAM WOULD LOOK. [U][N]OW WE HAVE SHIFTED INTO ITALIC[L].



Unhappily, the situation does not end here. The reader may have noticed that two codes may also be appropriated to introduce still another condition: *upper magazine* and *lower magazine*. This presumes a linecaster which is a *mixer* and which can, upon signal, automatically shift from one magazine to another, thus permitting 91 alternate channels, each containing two-character matrices (91 each in the upper-rail and lower-rail modes).

Let us suppose that the alternate magazine offers bold and bold italic, and let us use "[LM]" to represent the basic or lower magazine and "[UM]" to represent the alternate magazine. Hence we can now have:

[LM][L][C]R[S]oman. [U][C]*I*[S]*talic*. [UM][L][C]**B**[S]**old.** [U][C]***B***[S]***old*** [C]***I***[S]***talic.***

We may now see that there are in fact eight different conditions available to us under TTS hot metal coding: four "fonts" with an upper and lower case mode available to each. To clarify further:

Basic Magazine	*Alternate Magazine*
Lower rail	*Lower rail*
Upper case	*Upper case*
Lower case	*Lower case*
Upper rail	*Upper rail*
Upper case	*Upper case*
Lower case	*Lower case*

Review Questions:

1. What do the initials "TTS" stand for, and what is the meaning of the term?
2. What are "wire services," and how do they relate to TTS?
3. Why do you suppose there was initial union resistance to the transmission of pre-justified stories?
4. What was the basic adaptation which had to be made to linecasters in order for them to accept and use TTS tape?
5. Describe the origin and significance of "unit count" fonts.
6. What is a "multiface perforator"? How did it differ from the standard TTS perforator and what were its most common applications?
7. Describe the various improvements that were made in paper tape coding following Emile Baudot's development of the five-level telegraphic code.
8. What was the rationale behind the assignment of particular "holes" to represent certain characters?
9. Describe the manner in which a "precedence" code functions.
10. Why does TTS coding need to employ two different kinds of precedence codes? What about three?
11. When would it be necessary to employ the precedence code corresponding to "upper magazine" and "lower magazine"?
12. Can you think of arguments favoring the standardization of "upshift" allocations above the numbers? What arguments could be marshalled against such standardization?
13. What is the economic justification to be found behind the TTS process of setting type—in wire service applications or in a non-remote commercial environment?

7

The Beginnings of Photocomposition (First-Generation Machines)

IN THE YEAR 1946, two events of significance marked the beginning of the era of photocomposition: one was the first field testing of the Intertype Fotosetter, and the second was an agreement between the Lithomat Company and Higonnet and Moyroud which led to the development of the Photon typesetter.

The Intertype Fotosetter was first exhibited in the year 1950. It, as well as the Monophoto (which came along shortly thereafter) are what we term "first-generation" phototypesetters. By this we mean that, although they expose type images on photographic film or paper rather than to cast type or slugs from molten metal, in their basic design they do not differ substantially from their hot metal progenitors. On the other hand, the Photon typesetter, which was developed at about the same time, was a true "second generation" machine because it did not in any sense resemble a hot metal caster, being based upon entirely new electro-mechanical principles.

The need for a photocomposition device was readily apparent. There had already occurred a shift in the printing process from letterpress to offset. While letterpress required printing directly from raised type, or from stereotype or electrotype plates formed from an impression of raised type, offset—being a planographic process—required exposure of the plate from a photographic negative or positive[1] and the use of raised type for typesetting was, under the circumstances, a liability. In fact, elaborate techniques were devised to provide either the equivalent of a photographic positive from metal by printing "reproduction" (or "repro") proofs directly on a transparent carrier, or by rubbing the type form with carbon black, then burnishing the surface, and photographing the form itself. This latter process was known as "Brightype."

In addition, in the late 1950s, raised printing plates created by photographic means began to appear (*e. g.*, magnesium or acetate "etched" plates, or photopolymer plates such as DuPont's Dycril). This technology set into motion a

1. Whether a negative or positive is desired depends upon whether the offset plate is "negative working" or "positive working." With a negative working plate, that part of the coating which is not exposed to light is soluble in the developer, whereas with a positive working plate, the coating which is exposed to light becomes soluble.

flurry of experimentation which—while it did not deter the growth of offset printing—made its own contribution toward the ultimate obsolescence of hot metal typesetting.

Among other factors contributing to the growth of offset was the great expense of producing letterpress engravings (both line and halftones). The offset process eliminates completely the need to etch illustrations into zinc and copper plates, to mount these plates on wood blocks, and to lock these blocks into a metal form for imposition or platemaking. And even before the advent of "photocomposition," display heads were being composed by such devices as Filmotype, Fototype, Hadego, Typro, and other photo-lettering machines. These served the same general purpose as did the Ludlow process of casting display slugs from hand-set matrices. But of course for letterpress they required that cuts be made to provide the required raised image.

The (Harris) Intertype Fotosetter

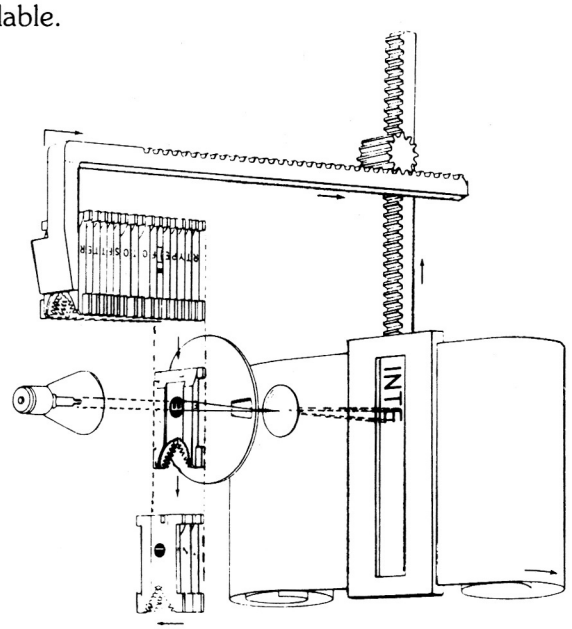
Since the Intertype Company was subsequently acquired by Harris—(becoming first the Harris-Intertype Company and then, more recently, the Harris Corporation)—we shall associate the name "Harris" with the product in our discussion, even though there was, at the time of the Fotosetter's introduction, no such connection. The Fotosetter development began toward the end of World War II, and the first units were tested as early as 1946 in the U.S. Government Printing Office. However, the device was not shown to the general public until the year 1950, at the first major printing exhibition to occur in the United States following World War II.

The Fotosetter was a keyboard-operated machine strongly resembling its hot-metal counterpart. It utilized the slug-casting machine principle of the keyboard cam-release, and the hot metal technique for the assembly of matrices, as well as for their distribution back into the magazine. But the actual act of "casting" was replaced by the substitution of a camera, and by the introduction of certain spacing and justification features.

The matrix which contains the photographic image is shaped precisely like hot metal circulating matrices, including the teeth and notches which guide the matrix back into its proper channel. The thicknesses of the brass matrix, as with a hot metal mat, determined the amount of space the character was permitted to occupy within a line, but the process worked in a different fashion. The illustration at the top of the next page indicates the principle involved. The images are stationary as they are photographed, but the film itself, including the elaborate mechanism for supporting the film cassette, moved, after each image was photographed, by an amount corresponding to the width of the matrix. Any one of eight pre-focussed lenses could be inserted (with appropriate adjustments in the carriage-movement mechanism) and two sizes of Fotomats (8 and 12 point) made it possible to offer selected settings between 6 and 36 points. Other lenses could be obtained to set 4, 5, 16 and 30 points.

Initially, each mat contained only one matrix, and so the number of channels was increased to 117.

Justification.—There were no spacebands. The camera mechanism measured the height of the pile of Fotomats after the line had been assembled, and the difference between the height of the pile and the desired line length was automatically distributed throughout the line. Thus the line was justified only by means of letterspacing—that is, assigning an equal amount of the left-over space to each character in the line, including interword spaces. Later models of the machine eliminated this deficiency and offered a capability of adding extra space only between words. A "Fotomatic" model was also developed, driven by TTS tape, and a two-letter Fotomat was introduced. A turret lens system, offering eight sizes, was also a feature of the Fotomatic, although size changes could only be accomplished by a manual movement of a dial, and not by tape command. However, the Fotomatic did not make the impact upon the market which had been anticipated since, by the time it was offered for sale other (second-generation) typesetters had become available.



The Fotosetter's principle of escapement. Note how the gear rack determines the manner in which the film carriage moves downward in order to make room for the positioning of the next character.

The Monophoto

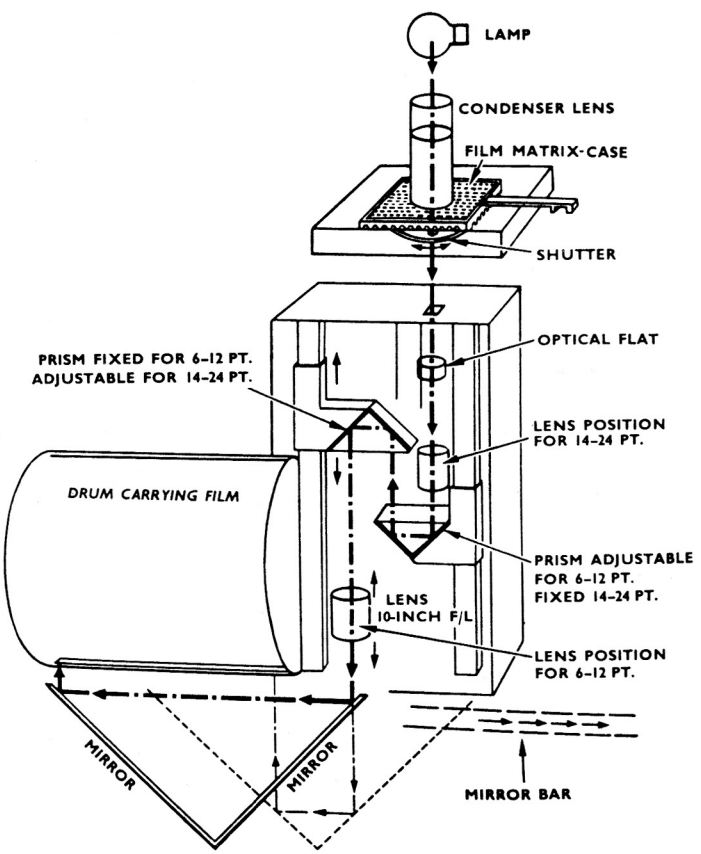
The other first-generation phototypesetter was the Monophoto, which was originally developed by (British) Monotype's George Westover as the *Rotophoto*, in 1949, but it subsequently emerged (in 1955) as the "Monophoto." It was based upon the same principles as the Monotype and hence it, too, is a first-generation machine.

The keyboard unit prepares a paper tape, justified at the keyboard in much the same manner as with the hot metal Monotype process. This tape then drives the camera unit which originally contained a 15 x 17 mat case. Later, in 1963 it was changed to 16 x 17. This mat case moves vertically and horizontally in the same fashion as with the Monotype, in order to position the desired character at the location where—instead of providing a mold for casting—it furnishes an image

for photographing. A single lens, which operates more or less on a "zoom" principle, sizes the image in a range from six to 24 points—although manual adjustment is required to secure the desired magnification. Several different sizes of photo matrices were available.[1]

The film, although it is of course moved forward in front of the "camera" for the setting of each successive line, does not, like the Fotosetter, move horizontally across the line on a character-by-character basis. Instead, a set of mirrors steps across the line, in order to "lay down" each character according to its indicated width. That width is ascertained from the character's location, in the same manner as for hot metal, and from "knowledge" of the character's width the data are provided to govern the movement of mirrors much as it governs the wedges of the Monotype.

1. In 1965 the Mark 3 version was introduced, offering an increase in running speed and improved justification. In 1967 the Mark 4 increased the matrix case to 340 characters and spaces, and in 1969 the Mark 5 Monophoto filmsetter appeared, offering a much greater flexibility of film feed for variations in interlinear spacing.



The optical system of the original Monophoto. Note how the illumination is brought to bear upon the character in the mat case which is positioned over the exposure aperture. And note, too, the sizing lens, and the pair of mirrors used to provide for character escapement. There are certain similarities between the use of the mirror escapement principle here and that subsequently developed by Photon for its Pacesetter models. (See page 107.)

The quality of the image of the Monophoto was and is superb, but the speed of the phototypesetter did not represent any significant improvement over its hot metal counterpart.

The ATF Typesetter

There is perhaps one other "first-generation" phototypesetter, and that is the "ATF"—a product of American Type Founders—most notably the model known as the ATF B-8. Whereas the other two first-generation devices were derived from hot metal products, the ATF typesetter was copied almost precisely from the Friden Justowriter—a strike-on machine. The Justowriter had first been introduced at the 1950 show by a company called Commercial Controls.

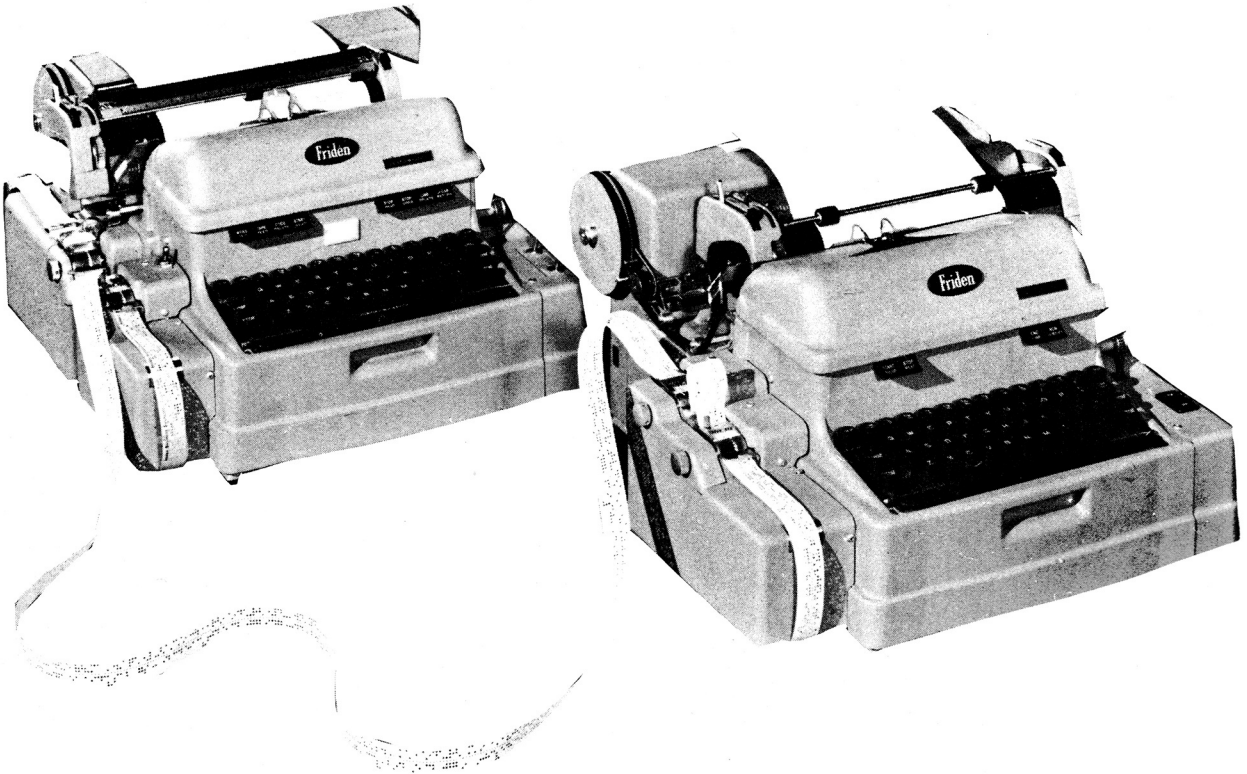
It may be appropriate to digress for a moment to describe the background out of which the Justowriter emerged. In the year 1948, a wave of composing room strikes occurred in the United States. The primary reason was the passage of the Taft-Hartley Act, and the reaction to the new law by the International Typographical Union. In the summer of 1947, that Union's convention initiated a strategy recommended by its president, Woodruff Randolph, that "reserved to its members" the right to strike, automatically terminating their labor contracts, if employers sought, by the hiring of non-members of the Union, to comply with the mandatory provisions of the Taft-Hartley Act (which outlawed so-called "closed shops").

Not since 1921 had there been such a wave of strikes, and the effect of these dislocations—in some instances the strikes lasted as long as six months—strengthened the resolve of the employers (particularly in the newspapers) to decrease their dependency upon the skills of their Typographical Union members. Moreover, many newspapers experimented with substitute processes during the period of the strike, and these processes were, perforce, "strike-on" or typewriter-like solutions.

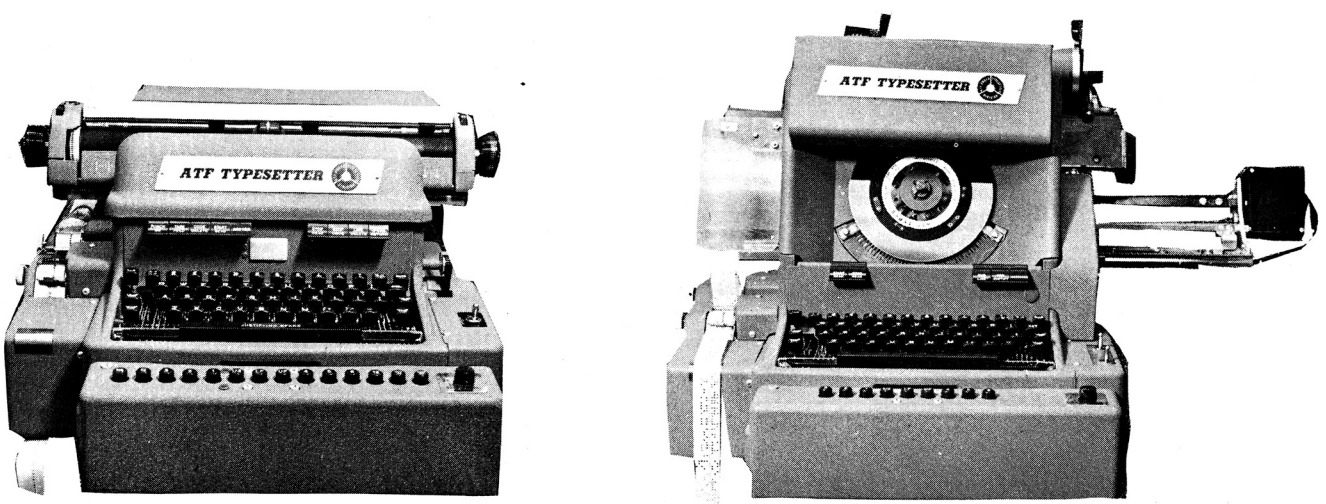
The Justowriter emerged in response to this expressed need, although as a matter of fact, it never did make any inroads in the unionized sector of the industry. However, it found its way into the small weekly newspaper field, and became the principal means by which such newspapers were "typeset" in the period of the late 1950s through the 1960s. Such papers were increasingly printed by offset, especially relatively low-cost high-speed web offset presses.

The strike-on methods of composition which were developed made use of such machines as the Coxhead Composing Machine DS-J, the VariTyper, the Fairchild Lithotype Composer, and the IBM Electric Executive Typewriter, but the Justowriter led all the rest in terms of sales and general acceptance for run-of-the-mill strike-on "composition."

The Justowriter consisted of two machines: a *Recorder* and a *Reproducer*. The Recorder produced a preliminary "hard copy" or typed sheet of paper as well as a perforated paper tape. Operators made their own end-of-line decisions when a light indicated that the text was now within justification range. Width values were ascertained by assigning to the characters one of four set widths, from two to five relative units. Hence, a "Justowriter" font is a unit-count font in the sense that all like characters—whatever the type face design—have a uniform width. The



The Friden Justowriter Recorder and Reproducer Units. These "strike-on" machines provided the chassis and the inspiration for the ATF typesetters.



The ATF B-8 Recorder and Reproducer units. Note that the Recorder unit is much the same, with a prominent light signal which is turned on when the point is reached after which the line will justify. You are encouraged to make your own end-of-line decisions shortly thereafter, before you go into "overset." The Reproducer unit of the ATF provides a rotating film disk and a moving take-up cassette and film transport instead of the striking bars and moving platen of the Justowriter.

Reproducer is capable of setting only one size of type—that machined onto the striking bars. When the operator terminates the line, keying the justification code, this code (when encountered by the paper tape reader on the Reproducer unit onto which the tape is subsequently mounted) informs the Reproducer as to the amount of interword space escapement needed in order to cause the line to justify. Hence only one typing is required.

Commercial Controls was subsequently purchased by Friden, and the major marketing of the Justowriter was carried on under Friden's auspices.[1]

1. Subsequently, in 1963, the Singer Manufacturing Company acquired Friden and continued with the sale of Friden products.

More than 19,000 Justowriter units were sold between 1951 and 1970—the year when the decision was taken no longer to manufacture this faithful old workhorse. But, in addition, somewhat more than 1100 machines (or the chassis of the machine) were sold to American Type Founders (ATF) for modification into phototypesetters, models A, B, B-8, and CS.

ATF left the input unit (The Recorder) much as it was, but modified the Reproducer so that, in place of the platen, a film carrier was mounted. And in place of the striking bars, a rotating disk was positioned, along with a light source. When the paper tape called for a particular character, the disk rotated forward to that location, stopped, and the light source then exposed the character so positioned onto the photographic paper. Then the platen moved over by the width of that character.

The disk offered 168 characters in one size only, consisting of two duplexed fonts of 84 characters each. Body sizes (on different disks) were available in a range from 5 to 14 points.

The ATF machines never made very much impact upon the conventional typesetting market, but they deserve to be included among the "first-generation" typesetters.

The Photon 200

We turn now to the development of the first "second-generation" typesetter —the Photon 200. As we have already stated, this machine was developed initially in the late 1940s, and was brought to the United States still in the conceptual stage by the inventors, two Frenchmen, Louis Moyroud and Rene Higonnet.

According to legend, at least, these two engineers had been doing research on the timing of airplane propellers by the use of high-speed "strobe"[2] lights. They prepared a report, describing their studies, and in their search for a typesetter for this article they were more or less appalled by the inconvenient and old-fashioned typesetting technology that they were exposed to. They began to think in terms of a spinning disk containing photographic images, with a high-speed timing device to keep track of each image as it went spinning by. When the desired character passed the strobe light source, a flash would be triggered, thus exposing the character "on the fly." "Escapement" was to be provided by a prism or mirror

2. "Strobe" refers to *stroboscopic*—a high-intensity, short duration light source which can enable the camera to take a picture of an image in motion as if in a state of rest.

arrangement that would advance across the film by the width of each character as it was to be exposed.

One could write a book simply about the trials and tribulations of Photon and its products. A company called "Lithomat" came to an arrangement with the inventors in 1946, subsequently changing its name to Photon. It sought financing from influential backers in the printing and publishing industry. It was first conceived as a quasi-non-profit venture, but after it had exhausted its initial infusion of capital the company was restructured as a publicly-held private corporation.

The first operating prototype (1948) is at the museum of printing in Lyon, France, and the early model of what was to become the Photon 200 line (a machine called "Petunia"—1951) is at the Gutenberg museum of printing in Mainz, Germany.

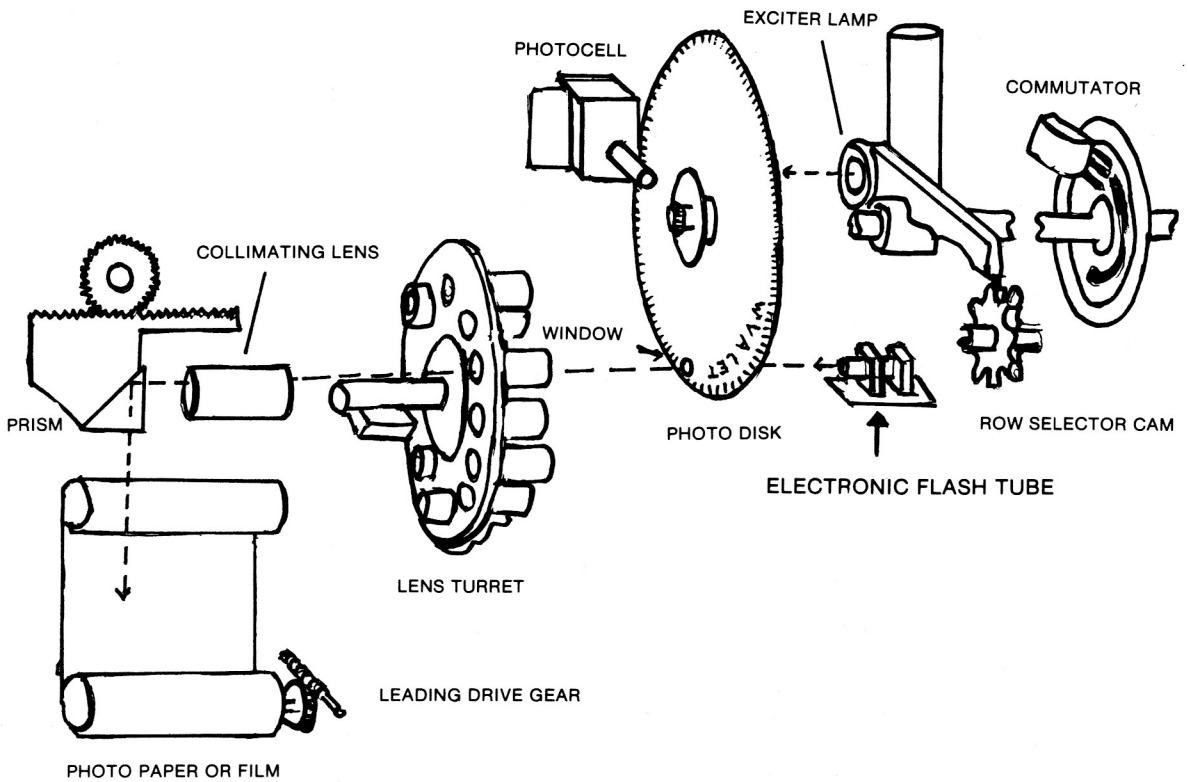
The Photon 200 consists of a keyboard unit, a second unit called the "relay rack," and the output unit—all of which work together in the manner of a "direct-entry" typesetter. The operator composes lines, and as they are released they are passed on to the output unit for typesetting. A typewriter is used as the keyboard, and it produces hard copy during the keyboarding process, as it sends signals to the relay rack and receives, in turn, a display countdown of the line length remainder and indicators of point size, leading and selected font. When the line has been composed and "released" it is set on the output portion of the photo unit. This box contains the film transport, a revolving disk with 16 fonts—each font occupying one-half of one of eight concentric circles of type masters. A lens turret with twelve lenses provides for selected point sizes, generally from four to 72 points.

The relay rack provides the computations and houses the width cards which are required for each of the various fonts. It then furnishes the output unit with the information which *it* requires as to the row and font to access, the character to flash, the lens to position, and the width to escape for each character and for each interword space.

Photon subsequently separated the keyboarding or input function from the output unit, and offered models which were driven from paper tape. These are the Photon 260, the 513, the 540 and the 560. All use the same spinning disk output unit. Each requires paper tape input of pre-justified lines. However, the 513 contains the logic to look up its own width values, which are "plugged in" by means of stored width cards. The 540 and 560 models, on the other hand, require that width values be provided directly on the paper tape along with the character identifications. This means that more frames of information—a "frame" is a row of holes across the paper tape—have to be provided for each character by the paper tape punch, and decoded by circuitry behind the paper tape reader. First the character is identified by position on the disk, and then one or more additional frames indicate what the width of the character is intended to be. The same is true of spacing codes: a space is called for and the width of the desired space is also stated.

Initially, the Photon operated on a 24-unit-to-the-em system, but the company then changed to a 36-unit system. Later models (all paper-tape driven) added still more capabilities, providing, for example, access to characters from either of *two* spinning disks. There is also a one-half sizing lens which drops in front of the lens turret, thus offering a total of 23 sizes from the same masters.

All of these models evolved from the basic Photon 200 concept. The spinning disk, along with the use of a high-speed xenon light source for flash exposure of the moving character images, represented a significant Photon patent (sustained on various occasions in subsequent litigation or out-of-court settlements). Another basic Photon patent involved the use of the "width plug" or "width card" as a means of storing escapement values.

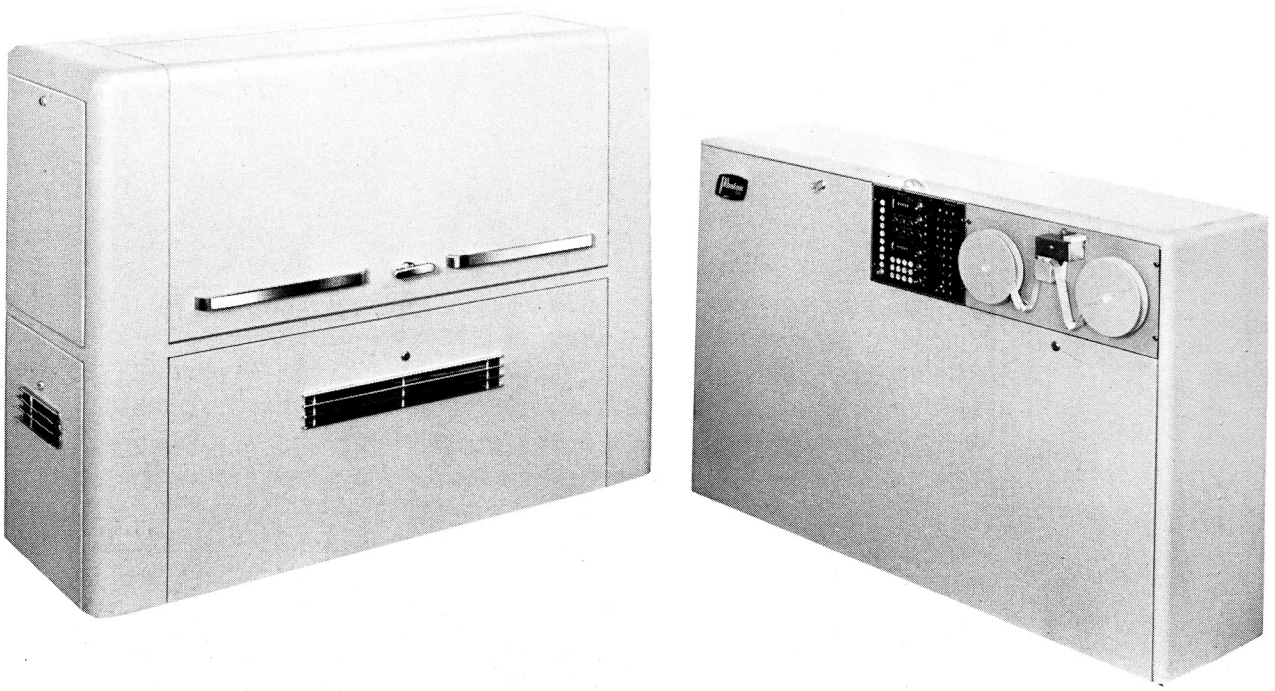


Optical Principles of the Photon 200

The Photon 200—optical track. The photo disk, containing 16 fonts in eight rows, is mounted on the spindle, and rotates continuously at a constant speed. Pulses from the exciter lamp enable the photocell to keep track of the timing slits on the edge of the photo disk so that the electronic flash tube will be triggered to expose the desired character. The negative image of this character (described as the "window") will pass through a sizing lens, a collimating lens, and be deflected by the prism onto the photofilm or paper. The prism "escapes" after each character has been exposed in order to lay down the next character image at the desired distance, determined by the width of the preceding character.



The Photon 200 "Admaster." You will note that this is a keyboard-oriented machine, with an elaborate panel of controls on the right to specify line length, leading, font, sizing, and other conditions. See also the picture on the next page of the 560 keyboard, which resembles this station.



The Photon 560 Display Master is driven from paper tape. The box on the right contains the paper tape reader, and relays that, in turn, direct the typesetting functions to be performed by the typesetter unit, housed in the box on the left.



The Photon 560 keyboard. Note how complex the keyboard is, with special keys for each face, each size, and each typographic function. As you can see, the device outputs punched paper tape.

The Mergenthaler Linofilm

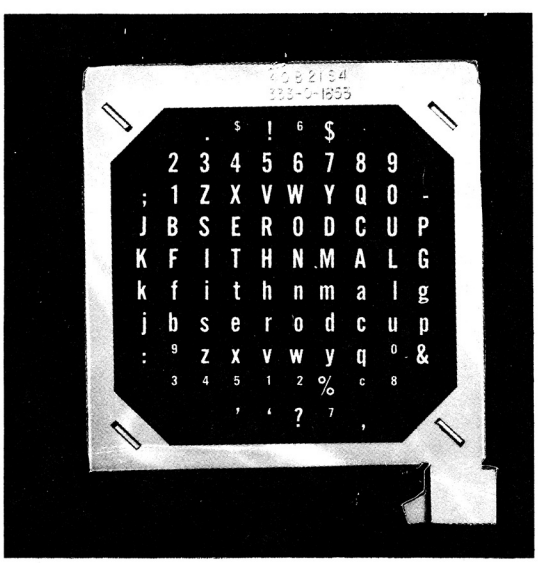
We shall not take the time to discuss each second-generation phototypesetter as it emerged. In fact, even today more and more of these devices are entering the marketplace—improved versions, of course, offering many new features, and selling at surprisingly low prices considering the cost of the early machines. But we should make mention of at least the next machine to reach the marketplace after the Photon. This was a second-generation device called the "Linofilm," and it was conceived and produced by the Mergenthaler Linotype Company in response to the Intertype and Photon achievements.

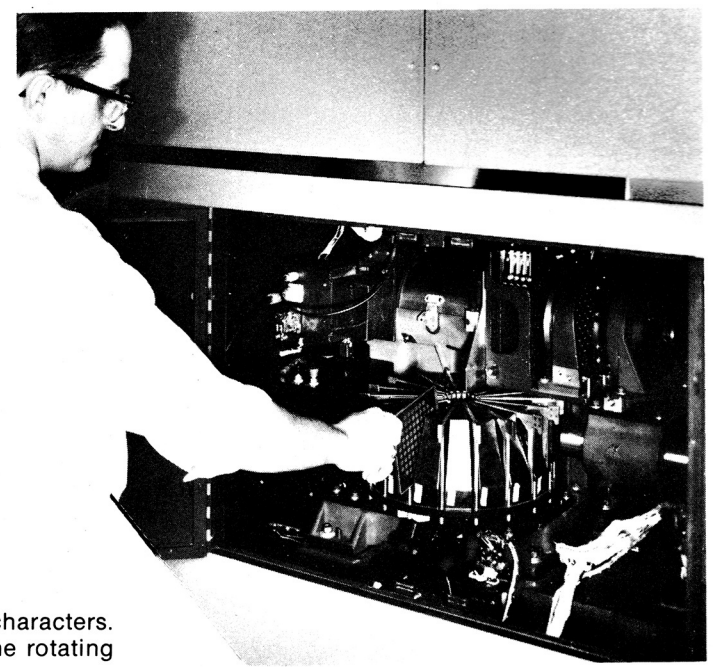
Mergenthaler had been caught short when Intertype came to the 1950 exhibit with a working typesetter, and Photon's prototype device was also well known. But back in the early 1940s Mergenthaler had done some experimental research with a cathode ray tube typesetting, and had abandoned the project as

□

untimely—as indeed it was. In this same period, Eastman Kodak had done similar research, as had RCA, but these were strictly laboratory experiments. However, for the 1950 show, Mergenthaler displayed a mock-up of a phototypesetter it had never built and wasn't at all sure it would in fact produce, largely to sample opinion on the subject of photocomposition. The reaction was sufficiently positive that they set about building a machine which reached the market place for field testing purposes in the year 1956.

The Linofilm separates the keyboard from the photo unit. Much like the Photon station (which was not so separated) the keyboard provided a continuous countdown as character widths were computed, and it also offered typewriter hard copy. Width cards to match the width values of the 18 fonts or grids that were to be mounted in the output or typesetting unit were positioned in the keyboard, and these were "consulted" by the keyboard's "logic." Changes from one grid to another were indicated by the use of supplementary buttons and switches on the keyboard.



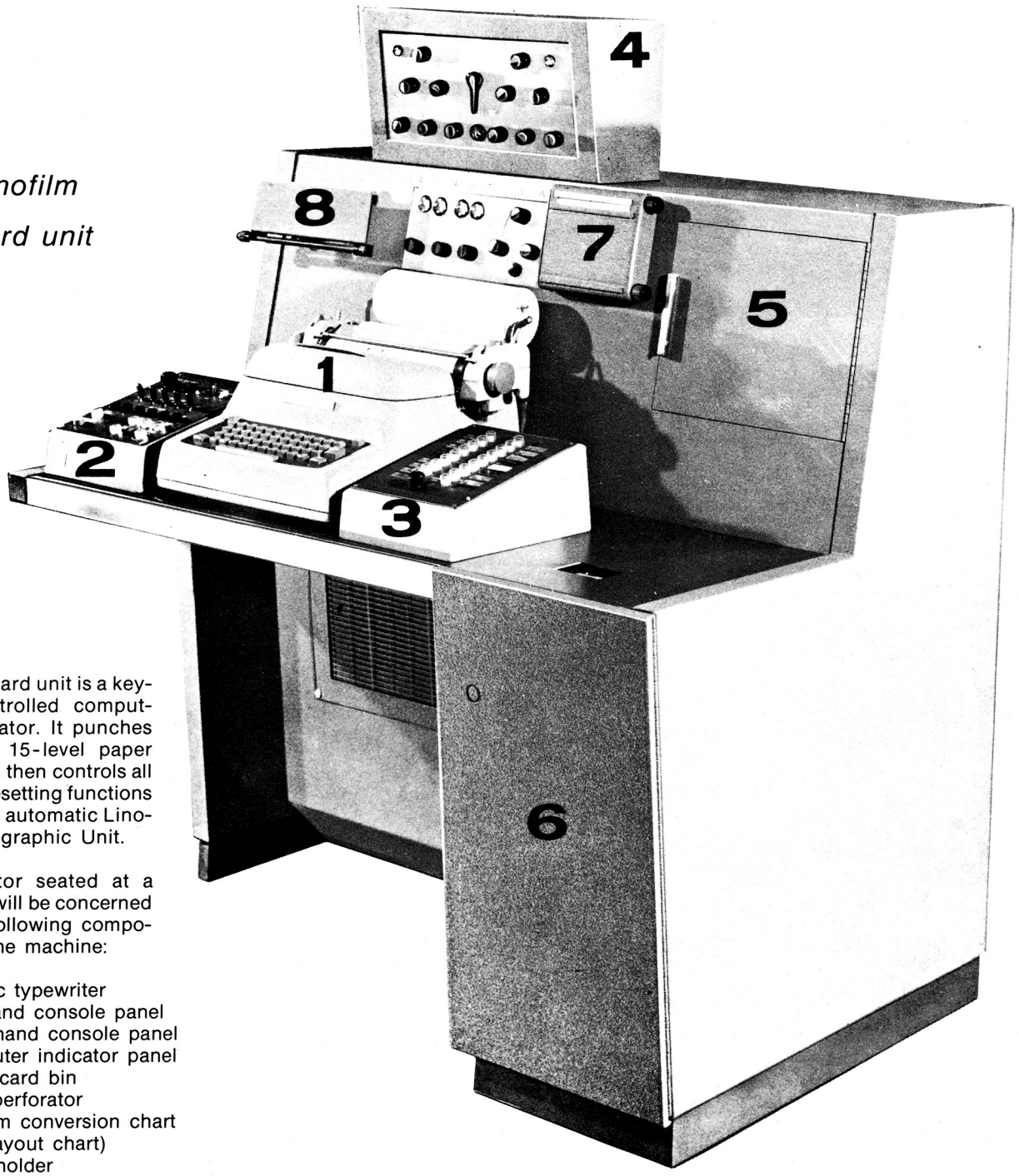


Above: The Linofilm grid contains a font of 88 characters.
Right: Eighteen grids are being positioned in the rotating grid basket.

The output unit contained a grid basket in which were seated 18 grids—each of 88 characters—and the desired grid was selected for immediate use somewhat in the same manner as a victrola record might be retrieved for play by a juke box. In other words, the grid turret revolved, and the desired grid was tipped out and held in place in front of the light source. A series of shutters masked out all characters except for the one that was to be exposed. This character was then sized through a zoom lens system, and a traversing mirror exposed the character onto the film plane. Thus the character image was stationary when photographed, and the quality of the image on photographic paper—or, more likely, on lith film, was superb.

In general, three different master sizes were offered: "A" range fonts were suited for text sizes; "B" range fonts were intended for display setting in the range from 12 through 18 points, and "C" range fonts were used for sizes from 18 through 36 points, although smaller and larger sizes could also be made available.

The Linofilm keyboard unit



The keyboard unit is a keyboard-controlled computing perforator. It punches a special 15-level paper tape which then controls all actual typesetting functions in the fully automatic Linofilm Photographic Unit.

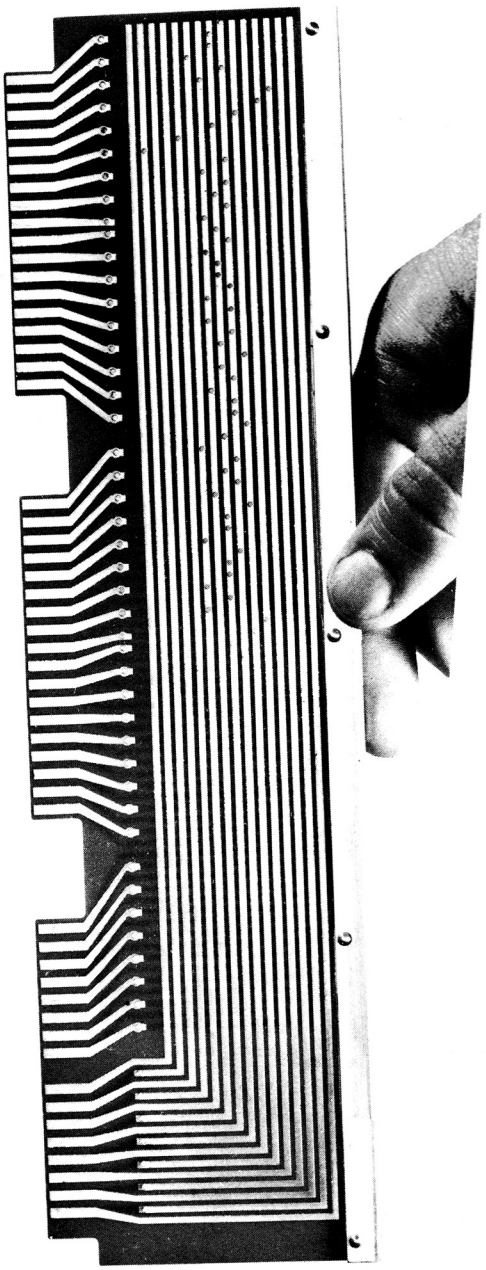
An operator seated at a keyboard will be concerned with the following components of the machine:

(1) Electric typewriter
(2) Left-hand console panel
(3) Right-hand console panel
(4) Computer indicator panel
(5) Width card bin
(6) Tape perforator
(7) Pica-em conversion chart (and layout chart)
(8) Copy holder

Because of the zoom lens method of character enlargement, different sizes could not be aligned on the same optical base line. But the use of different sized masters (in order to produce larger or smaller images through the same lens system) made it possible to select masters more suitable for the intended size ranges, with proportions more pleasing to the eye than is usually the case if one master image is to be enlarged or reduced over an extended range of point sizes.

The Linofilm output unit is completely a "slave." It is driven from a 15-level paper tape, considerably wider than the usual six to eight level tapes used by most other machines (except of course for the Monotype and Monophoto's "31-





Making a font selection from among the 18 available grids involves pushing the appropriate button. Obviously, width cards must be inserted to correspond with the grid numbers.

A Linofilm width card (shown at right) is mounted in the keyboard for each font.

level" ribbon). Fifteen levels were indeed required to convey the information about the character position to be exposed plus the width of each character. However, it was the output unit which calculated the amount of space to be inserted between words for justification purposes. The basic character widths were read from the paper tape by the first of two paper-tape readers. The second paper tape reader then deciphered the character calls and processed the information on a character-by-character basis, using the interword spacing values which, in the meantime, had been assigned to that line.

Summary

We have presented this historical approach to the development of phototypesetters in order to pave the way to a discussion of the more general principles that we must understand in order to trace the subsequent evolution of such typesetters. These principles, as they relate to "second-generation" photo units, are discussed in the next chapter. We shall reserve to Chapter 9 our discussion of third (and fourth) generation typesetting machines.

Review Questions:

1. What is the difference between a "first-generation" phototypesetter and a "second-generation" machine?
2. How did the general shift from letterpress to offset printing techniques contribute to the development of photocomposition devices?
3. How are character widths "read" by the Harris Intertype Fotosetter and by the Monophoto?
4. How is justification achieved on the Fotosetter?
5. In the case of both the Monophoto and the Fotosetter, character images remain stationary when exposed. How does escapement take place in these two devices?
6. How is character sizing accomplished by the Fotosetter and the Monophoto?
7. In what sense is the Friden Justowriter font a unit-count font?
8. What was the mechanism for exposing character masters on the Photon 200?
9. What was the function of the "relay rack" in the Photon 200?
10. What were the two fundamental technical advances patented by Photon as a result of their development of the Photon 200?
11. Describe the manner in which character images were displayed and exposed in the Linofilm photo unit. How does this differ from Photon's character exposure mechanism?
12. What is the nature of the "countdown" received by operators of either Photon or Linofilm keyboards?
13. Describe the difference character sizing techniques of the two units—the Photon and the Linofilm.
14. What do we mean when we describe a particular device as a "slave" unit?

8

Underlying Principles Of Second-Generation Typesetters

ON THE BASIS of our discussion of the evolution of second-generation phototypesetters, we can understand—even with only a preliminary introduction—that there are many different ways in which desired results can be achieved. It will therefore be helpful to consider at this time the fundamental principles that underlie all such devices. We shall discuss the following topics:

1. How characters are stored.
2. How characters are selected.
3. How characters are exposed.
4. How characters are sized.
5. How characters are laid down (the "escapement" process).
6. How justification takes place.

Character storage

We have already seen, in the case of the Photon 200 and 500-series machines, how characters may be stored by being mounted on a revolving disk. In the case of the Linofilm they are stored on a stationary grid. In all cases the characters are stored as negative images so that the light which passes through them will produce a positive image on the photographic film or paper.

Characters are placed very carefully on the grid, disk or other carrier so that they will not present problems of horizontal or vertical alignment. The "master" size of the character is not necessarily the size of the image on the grid or disk. The "artwork" which leads to the creation of the film images generally involves the following procedures:

A determination is first made as to what the "true" master size should be. For example, if the type is supposed to look its best as a text face a master drawn to look well at, say, ten points may be selected. This master will then be enlarged many times so that it can be touched up and perfected by hand. Then it will be

photographed to the size required by the typesetting machine's optical system design—the size at which the character will appear on the disk or grid. That size may be ten points, or five points, or even two and one-half points, or some other appropriate size. In other words, the machine design may be such that to set on a "one-for-one" basis (*i.e.*, an original ten-point master for ten-point type), a magnification factor of 2:1 may be contemplated. In such a case, the film master on grid or disk would turn out to be five points.

Whatever the stored master size may be, it will be mounted or otherwise presented on the carrier—a grid, a disk, or film strips to be affixed to a drum. In general, these characters will be arranged in "fonts" and the font may vary in repertoire from as few as perhaps 84 or 88 characters to as many as perhaps 120 or so. Usually—but not always—more than one font is available on the typesetting machine at one time.

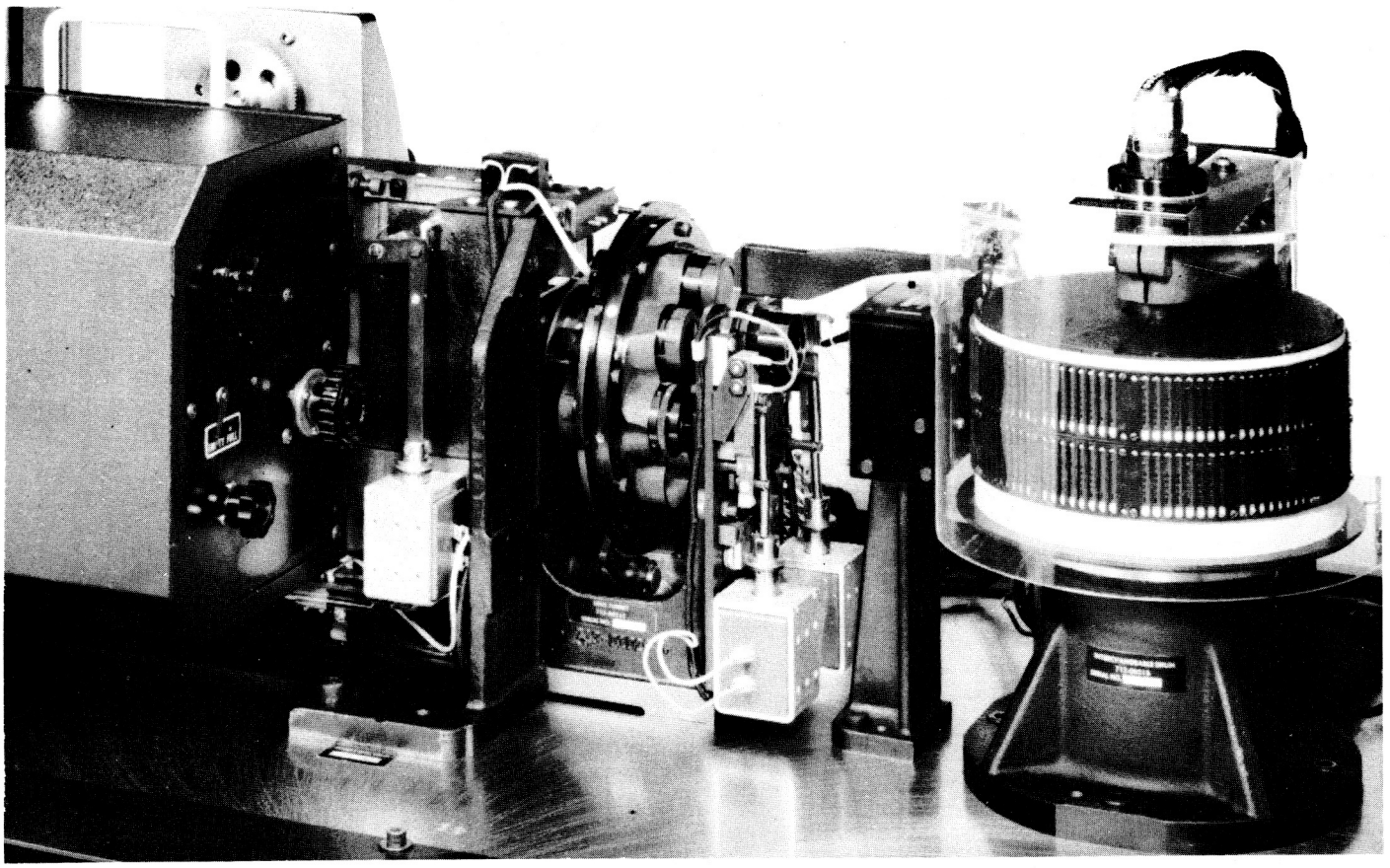
The manner of storing fonts—*i.e.*, whether on a grid, disk, drum, cylinder or otherwise—is determined by the method by which the desired character within the font or from among an array of characters comprising more than one font is to be *selected.*

Character selection

There appear to be two basic principles for character selection, although both principles may be at work more or less simultaneously. One is to position a particular character over the light source (or the light source over a particular character) and to flash the light at that time. The other is to expose all or a number of characters and to mask out the images of all characters except the one that is intended. It is possible, for example, as is the case with the Linofilm, to illuminate the entire grid, and then to mask out all characters except one. It is possible to move the individual character in front of the light source (as with the Monophoto) or as with the ATF B-8 where a disk is rotated and then halted in place at the time of flashing, or as with the Photon 200, where the character is exposed "on the fly" by careful timing of the flash. Let us look more carefully at a few examples:

Consider the illustration of the Photon 713, shown on the next page. The light source is located inside the spinning drum. In fact, there are two light sources. One exposes simultaneously all four characters in the same drum position for the upper four fonts. The other exposes four characters in a given drum position for the lower four fonts. In other words, the first step in the selection of the character is to determine the drum position. The second step is to determine whether the desired character is to be found in that position on the film strip which contains the upper four fonts or in that position on the film strip which contains the lower four fonts. If the former, the upper xenon flash would be triggered. If the latter, the lower xenon flash would be triggered. In either case, four characters are simultaneously exposed. But there is a shutter which moves up and down behind the row of characters, which masks out all rows except one in the upper matrix of four fonts and one in the lower matrix of four fonts. When

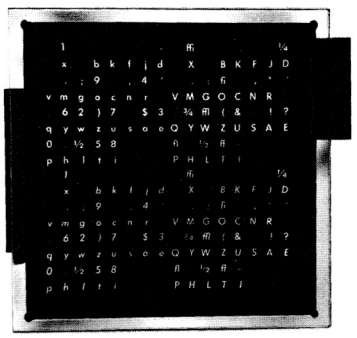
the shutter is in position to let light through to expose *row one* of the upper font "pair" (if the upper light source should be flashed), it would also let light through to expose *row one* of the lower font "pair" if the lower light source should be flashed. Here, then, we can see that character selection occurs both by moving certain characters into an exposure position—in this instance while they are revolving at a constant and relatively high speed—and also by masking out the unwanted characters that would also be exposed from that position.

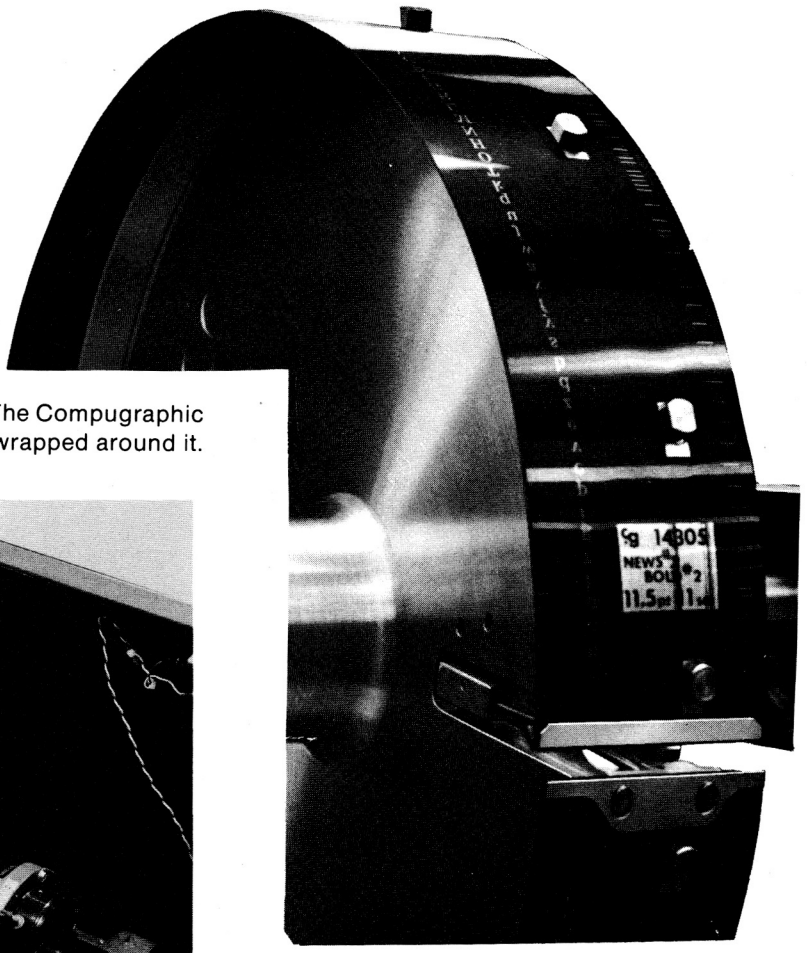


An interior view of the Photon 713. On the right is the font drum, onto which are mounted two film strips, each with four fonts. Wires attached to the xenon flash units can be seen rising out of the center of the drum. These units are located inside the drum itself. Immediately to the left is the shutter which masks out the unwanted characters. Next comes the revolving turret, into which are mounted eight sizing lenses. At the top left of the picture is the film or paper cassette (with handle) which contains both the unexposed and the exposed medium. When the cassette is mounted, a shutter at the film plane location is opened and remains open until the cassette is dismounted for processing.

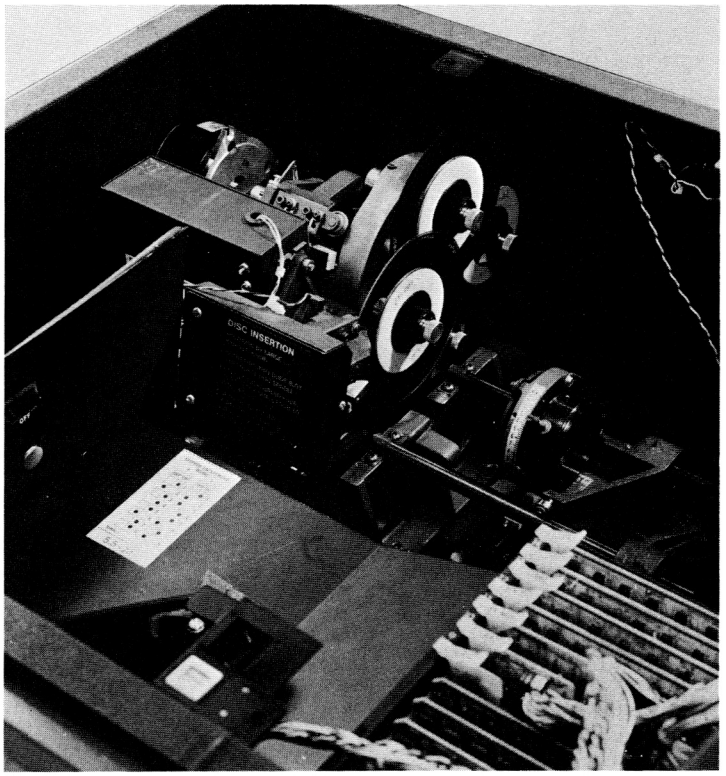
In the illustrations on the next page, all characters that are exposed (as well as those that are masked out so that they cannot be exposed) are located in the same optical path. But it is also possible that all characters may *not* be located in the same optical path. Consider, for example, the design of the Photon 900 (or "ZIP" as it was called because it set type very fast and in both directions—from left to right and from right to left). This machine normally contained

three large grids of 88 characters each, providing a total of 264 characters. There was a xenon flash tube *behind each character* so that the desired character was selected *merely by flashing one particular lamp*. However, in order to bring that character into the proper optical path, a reciprocating lens would





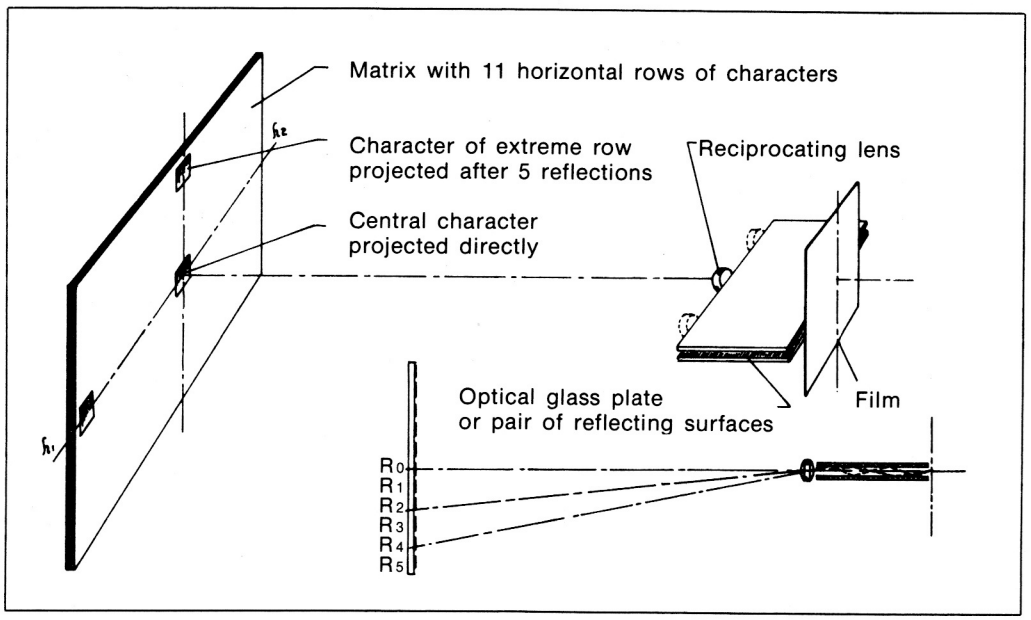
Above: A grid from the Linofilm Quick. *Right:* The Compugraphic 2961 font drum, showing how the film strip is wrapped around it.





Right: AM VariTyper's font disk used with the Comp/Set and Comp/Edit typesetters. Bar codes on outer rim contain character width information. The inner four rows of characters—each row a different font—are photographed for typesetting. *Left:* AM font disks mounted on a Comp/Set. This is a four-disk machine. The left disk is in position for typesetting. The turret can be rotated to bring any of the other disks into the extreme left position.

catch the image and direct it through a pair of reflecting surfaces, catching the light rays and straightening them out so that they could be passed on to the lens system and to the photographic film or paper. Characters positioned directly opposite the lens would pass straight through. Characters one row below or above this location would be bounced once from one mirror to the other and then passed through. Characters two rows above or below the set of mirrors would be bounced twice, and then passed through, and so forth.



Illustrating the basic principles of the Photon "Zip" or the 900-series machine. Imagine an array of light sources, one behind each character, in each of the 11 horizontal rows of characters. Notice how the reciprocating lens plates align the images at the level of the film plane.

Here, then, we have an extreme example—character *selection* by having a different light source for each character. This is, of course, an expensive solution to the problem of character selection, and one that offers many maintenance problems because the intensity of each light flash must be adjusted in relation to its distance to the photographic plane, and the intensity of each must be consistent (other things being equal) with the others, or otherwise the weight of character exposure will be uneven.

It would not be practical to have a large number of light sources in a small space, such as behind each row of characters on one font master disk, but nowadays fiber optic tubes are able to conduct light to a specific point from a more distant light source. Note how this works for the Photon (now Dymo) Pacesetter, where different rows or rings of a disk can be selected according to the light source as it is conducted to a very specific location by means of a light tube. An alternate way of bringing a particular row of characters into position in relation to the light source (or *vice versa*), would be to shift the position of the disk itself. When you want to expose characters in one row you shift the disk upward (or sideward to the left for example), and when you wish to expose characters in another you would shift it downward, or sideward to the right.

In neither of the cases we have just discussed would it be necessary to mask out unwanted characters from other rows.

We have by no means exhausted the techniques which are brought to bear in order to assist in the selection of characters. There are many ingenious solutions. In general, those which involve the fewest moving parts are most successful. If characters are to be masked by moving shutters, the shutters may tend to stick or to clip off portions of the images, and a time interval is required for the movement of the shutter. If the entire disk must be moved (as to position a different font row in front of the light source), vibrations may be induced, causing the machine to jiggle and to produce a less perfect representation. (We found, years ago, that we had to "program" a blank character—a "null"—immediately after a row change on a Photon 513 to give the machine a chance to "settle" after a disk movement had taken place.)

However the character is to be selected, the choice must be made between exposing images on the fly and exposing images that are in a stationary position at the moment they are called for. Many people believe that exposing characters while they are in motion produces images which are not as sharp and as well defined as would be the case if the character master is held stationary in front of the light source while being "photographed." A careful examination of the output from a rotating disk may show a slight "tail" or blur caused by the motion of the disk during the brief duration of the flash exposure. Even if this "tail" is observable on the film or paper, it probably would not be seen by the naked eye on the output when it is finally printed. And there are other aspects of character imaging which may pose more serious problems—such as character alignment and character escapement.

There are, therefore, other considerations which are more significant from the standpoint of the quality of the image than the fact that it may have been moving when its "picture was taken." The design of the type itself, the quality of the lenses, the critical *timing* of the flash if the image is indeed in motion, and the *accuracy of the character laydown* are all at least equally important.

Some fonts mounted on a rotating disk or drum are so mounted that an error in the timing of the flash will result in irregular positioning with respect to the *base line*. Characters which move vertically across the film plane could be photographed either too early or too late to align properly with respect to the base line. On the other hand, character *spacing* would be adversely affected by improper flash timing in those instances in which the images spin by in a horizontal direction.

Summarizing the methods of character selection for second-generation phototypesetters, we find that:

1. There are devices which illuminate all characters simultaneously, masking off all but one. (This is likely to be the technique employed where characters are stored on a stationary grid, as with Linofilm or Linofilm Quick. However, it is not the method used in the case of the Photon ZIP—where the image is also stationary but where each character on the grid has its own light source.) On page 94 we shall illustrate the "masking principle"

as it applies to the use of a grid, with all characters evenly illuminated. The Wide-Range Linofilm Quick or SuperQuick affords a good example.

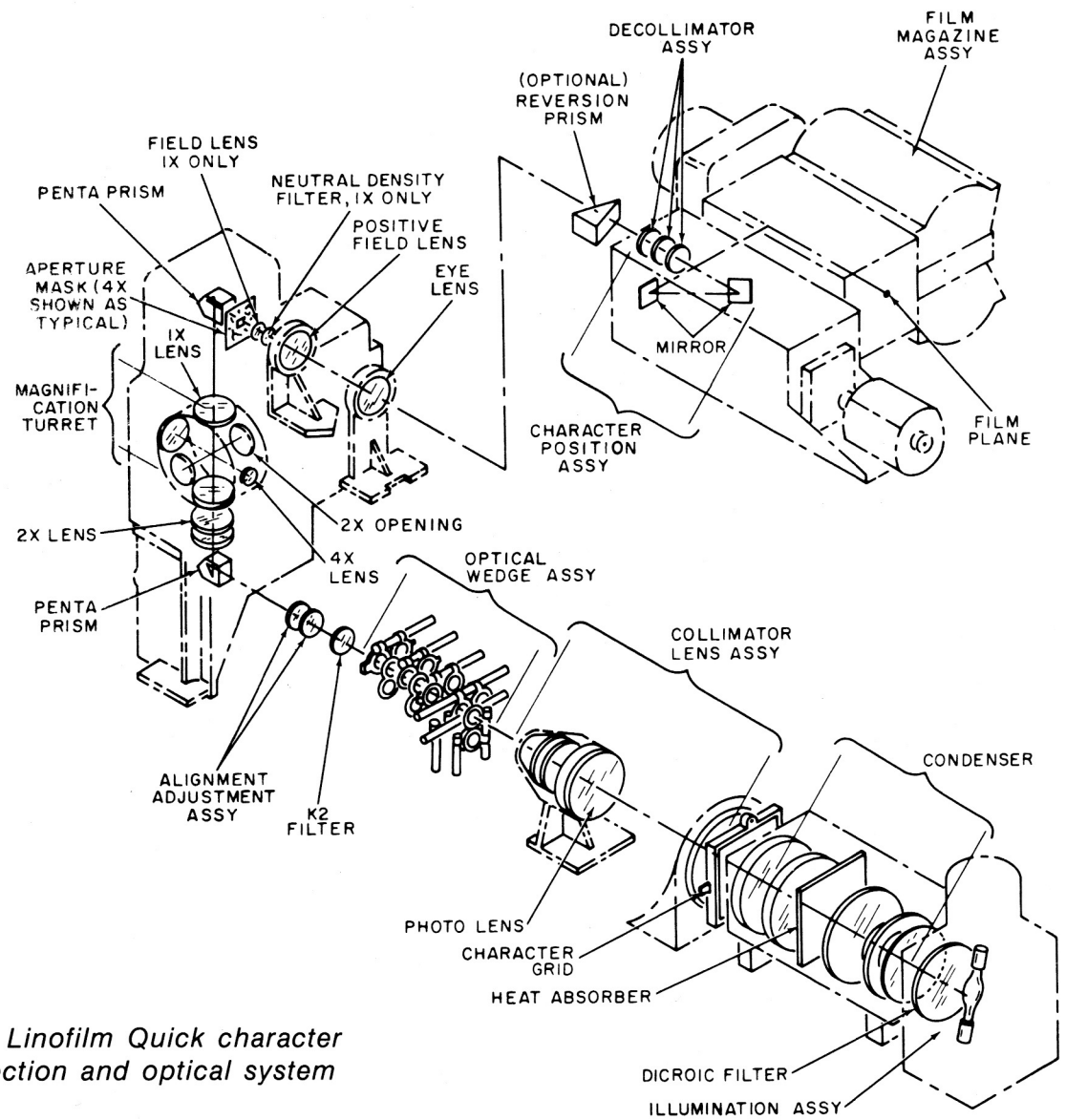
2. There are devices where only the selected character is illuminated, and that character is—at that moment—stationary. (This technique would apply, for example, to the ATF models, such as the B-8, and to the Mergenthaler V-I-P models, as well as to the Monophotos, Mark 1 through 5. In the case of the ATF, the spinning disk rotates and then stops. In the case of the V-I-P, the drum "oscillates"—moving in either direction, following the shortest path to reach the next desired character.

3. There are devices in which only the selected character is illuminated by the use of a fast flash "picking off" the image from a constantly rotating disk or drum:
   a.) and where the disk shifts in position in order to expose the desired font, or
   b.) where a different light source is used to select the character in the particular row which contains the desired font.

4. There are also devices in which more than one character is illuminated—but not all of them. Those illuminated but not desired are masked out. (This technique applies, for example, to the basic models of the V-I-P—except for the high-speed versions—and to the Photon 713 as well. However, in the former case the image is stationary when exposed and in the latter case it is rotating.)

On the next page we have selected the Mergenthaler Wide-Range SuperQuick to illustrate the method whereby one character is selected from a grid with 256 cells, by means of a masking process. The entire grid is evenly lighted, and then, according to the input tape signal, successive maskings occur until the image of but one character is permitted to pass through the lens system. Because the selected character may not be precisely opposite the lens, an arrangement is provided to deflect the character into a common optical path. The complexity of the masking procedures will no doubt impress the reader, for he may well imagine the number of mechanical parts which must be activated in order to select the desired character. The Linofilm Quick was a product of the late 1960s, and was very shortly superseded by the V-I-P—a device offering much greater speed and flexibility. However, even the earlier (*i.e.*, non-high-speed) models of the V-I-P also use a masking technique, although only a few characters, rather than all of those on an entire grid, are blanked out during the masking process.

In evaluating various solutions to the problem of the design of phototypesetters, the reader must be aware that some solutions are made necessary because competitors' patents may preclude the adoption of what would appear to be a logical answer to a specific problem. For example, the Photon patent on the use of a revolving disk, with timing slits, and a flash to "pick off" the character, has been case in point. Other manufacturers wishing to utilize this technique have had to pay royalties in order to do so—at least in some cases.



The Linofilm Quick character selection and optical system

How characters are exposed

We have already suggested that there are two basic ways in which characters may be exposed: from a stationary or from a moving position. If a grid is used, the character is always stationary at the time it is exposed, although the grid may move between exposures (as with many of the Monophoto models). On the other hand, the grid may not move at all: a light source could be associated with *each* character (as with the Photon ZIP) or *all* or *some* additional characters may be illuminated and all but one masked off. Some systems using grids may not require a high-intensity, short-duration light source, and such systems may be able to make use of slower exposure lithographic films because, as a rule, these systems operate at much slower setting speeds.

For the most part, however, second-generation typesetters operate at relatively high speeds, and although the flash is fairly intense, it is of such short duration that the film emulsion must be quite fast. Special "chemistry" is also required to develop these high-speed films or papers.

How characters are sized

In some cases, characters are not "sized" at all. They derive their size exclusively from the size of the image on the film strip or grid. This does not mean that the image may not be enlarged. The character may be stored and exposed as a smaller image than that which appears on the film or paper. But the lens system magnifies at only one ratio. On a one-lens Compugraphic 2961, for example, the lens enlarges at a ratio of 1.2X. If you want to set a particular size you must *mount the appropriate film strip.* You change the film strip and not the lens adjustment.[1] Thus one possibility is that characters will derive their sizes from the dimensions of the stored master and not at all from the lens system. It is even possible to offer more than one master on the same film strip or disk. For example, the Compugraphic 4961 offers four fonts, of which two may be of a different size. The Fototronic 600 offered six fonts, each of which *may be* of a different size.

1. Film strips are available from 5 to 12 points. A twin-lens option would convert these to 10 to 24 points.

Obviously, machines which do *not* do their sizing through lens systems are less flexible than those which permit you to enlarge or reduce from one master size. The former are often intended for such "straight matter" as newspapers or periodical composition, and if you require much "mixing" you will encounter the need for a great deal of hand stripping. If you want sizes other than those which comprise the current "dressing" of masters on the machine, you must stop the machine and mount other film strips. (With some such typesetters you cannot even change the film strip without exposing the output and spoiling the work!)

On the other hand, it is possible for such machines—even the inexpensive ones—to offer a high-quality type image, since each size can be designed to different proportions than all other sizes, and since more attention can be given to the design, focus and alignment of the lens system, as well as its quality. Only one —and not an entire set—of lenses is required, and it may be mounted more or less permanently. The Alphatype AlphaSette is one phototypesetter noted for the quality of its product, and it is generally operated with only one master size available at a time.

There are other instances in which you may use a single-lens system, but you may obtain other sizes by making manual changes in these lenses. Either you may unscrew one lens and mount another in its place, or you may throw a lever or rotate a turret by hand to position a different lens and thus obtain another size. For example, the Compugraphic CompuWriter provides for a 1.2X and a 2.4X lens system. Hence if you mount an eight-point film strip you can set eight-point type, or open the machine, blank out one lens and uncover the other, and set 16-point type.[2] If you mount a ten-point master you can thus obtain either ten or twenty point type. But the sizing requires manual intervention: either you

2. Some models, such as the CompuWriter II, permit the use of two point-size masters on the same film strip.

change the film strips or you change the lens (or perhaps you do both). But whatever you do, you must do manually.

Happily, most phototypesetters nowadays offer the ability to size type automatically, *i.e.,* without having to stop the machine and make manual adjustments. Signals to alter the type size may be initiated directly from the machine's keyboard, recorded on and conveyed from paper or magnetic tape, or from an on-line computer. This generally means, then, that one font master will be "sized" over a range which may vary from perhaps six to 72 points. However, when you read the literature provided by the manufacturer and learn that the machine offers, say, eight fonts in a range of from six to 72 points, you must also inquire as to the number of discrete sizes available on the machine at one time. If sizing is effected by rotating different lenses into position, then the number of such lenses will be stated. A "16 x 16" typesetter is one which offers 16 fonts in 16 different sizes. If sizing is effected by the use of one zoom lens system, the number of positions at which the lens may be arrested will determine the size combinations available.

We have seen, then, that you can "size" by changing the film master image, or by introducing varying lenses or lens positions. But you can also size by doing both. Consider, for example, The Mergenthaler Linofilm. The "A-range" master grids on the Linofilm can be enlarged or reduced by the zoom lens system, which racks itself to positions which provide (normally) 6, 7, 8, 9, 10, and 12 point type. The "B-range" masters can be enlarged or reduced (by using the same positions) to provide 12, 14, 16, 18, 20, and 24 points. The "C-range" masters offer 18, 24, 30 and 36 points. (Note that the A and B fonts offer six size variants to the "C-range's" four.) With certain restrictions, additional sizes can be produced by the use of extended grid ranges—"D" (42 and 48 points) and "E" (38 and 54). The zoom lens system employed by Mergenthaler on the Linofilm *did not permit different sizes of type to be base-aligning.*

The principle of using different-sized masters to set type within certain size ranges has been carried over to the Mergenthaler V-I-P. Certain models of the V-I-P permit you to use either size range "A" or size range "B" fonts. The "B" fonts are twice as large as the "A" fonts and take up twice as much room on the V-I-P drum. (A one-drum machine can accommodate either: (1) six "A-range" fonts, (2) three "B-range" fonts, (3) four "A-range" and one "B-range" fonts, or (4) two "B-" and two "A-range" fonts.) Again, the V-I-P uses a zoom or rack lens system, but in this instance the lens system is such that all sizes produced from the "A" range masters are base aligning (although A and B faces cannot be intermixed on the same line without considerable "fiddling"). This common alignment is achieved by not using the entire lens area, but only one-quarter of it, so that the character to be magnified is referenced with its base line in the center of the lens.

To continue with our example, Mergenthaler also offers models of the V-I-P which operate only from "A-range" fonts, and provide an extensive array of sizes from such fonts. The V-I-P offers ten different focusing positions, *but it also offers a supplementary lens*—either a 1½X or a 3X sizing lens which flips into position behind the zoom lens in order to provide further magnification. This allows the

"A-range" fonts, which could only be sized up to 24 points, to be enlarged to 36 (1½X) or to 72 (3X). Thus, for example, you would have available, ordinarily, the following sizes:

A Range	*B Range*
6	18
7	21
8	24
9	27
10	30
11	33
12	36
14	42
18	54
24	72

The zoom or rack lens systems we have considered so far have offered relatively few "stops" or sizing capabilities within the range of sizes possible. Until recently, the V-I-P has offered only ten sizes, from six to 24 points. Yet it is perfectly possible to design a machine to provide more "way stations" within the 6–24 point range. The typesetter being used to produce this book (the Comp/Set 4510) also uses a zoom lens system. Its normal range is from 5½ to 74 points, the first half of which are illustrated below. Seventy different sizes are available: 5½, 6, 7, 8, 8½, 9, and every point size thereafter, except for 55 points (since the command for that size would conflict with the command for 5½).

ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890!

Lens Turrets.—Lens turrets, on the other hand, offer only a certain number of sizing alternatives, and, in theory at least, they are less flexible than the zoom approach. You may have a typesetter which provides positions in its turret mount for only four lenses, or eight, or perhaps 12. The Photon (Dymo) Pacesetter has carried this all the way up to 16 possibilities by departing from the conventional turret design and mounting lenses along a spindle in different positions. And, as we have already mentioned, a secondary lens system could be implemented with turret systems (as well as zoom systems) to further enlarge (or reduce) the possibilities otherwise offered. This is the case with the Photon 532, and also with the Graphic Systems C/A/T and the Singer Photomix 8400, as well as the Mergenthaler V-I-P.

Where separate sizing lenses are provided, the user may be able to select which point sizes he desires (as, for example, whether he would prefer a six- or seven-point lens for his smallest size). In some instances he might even be able to purchase a lens which deflects characters onto a different base line. Hence he may be able to obtain a lens which can create six-point images above the base line as though they were true superiors, blending with, say, a ten-point text face. Occasionally one encounters the use (in a turret position) of an anamorphic lens which "squeezes" the width of type characters to make them more condensed and hence to fit more such characters into one line.

One other observation may be made with respect to the sizing of characters from a given master. When you increase or decrease the size of the image you may alter the intensity (or density) of that image. It may be necessary to accomplish (automatically) a mechanical change in the lens aperture to compensate for this problem, or to add filters to certain lenses in order to reduce the apparent blackness of the image at certain sizes. Alternatively, some typesetters are "programmed" to double or triple-flash in order to provide more light for larger point sizes, where the degree of magnification is so great that the character might otherwise appear to be washed out. It seems to be true, also, that the larger sizes should look blacker; but it is equally true that very small point sizes—such as six point—seem too light, and some of the serifs tend to disappear:

This is in part because the type masters used were probably not designed to look well at six points. The text type was probably designed to look its best at around ten points, and the serif strokes, when reduced to six points, may seem to be too delicate. On the other hand, a face designed from a ten point master might look very gross if enlarged, say, to eighteen points, as

we now illustrate here.

But quite apart from the design of the type itself, the problem of how to obtain the appropriate weight from each size remains. The typesetter we are using provides an arrangement whereby the flash intensity varies as the zoom lens moves back to set larger sizes. Hence more illumination is provided because the lens becomes "faster." The earlier model opened up the lens for larger point sizes, by means of a mechanical shutter contrivance. Students of optics will recall, however, that the depth of focus suffers as the lens aperture is enlarged, and sharpness of the image may be less assured.

Character escapement

At the outset, we drew an analogy to a typewriter and pointed out that with the conventional typewriter the platen moves by the width of each (monospaced) character after its key has been struck. If the typewriter is designed to "set" at ten characters to the inch, the platen moves to the left by one-tenth of an inch, and the gear track is so machined as to make this possible. Consequently, escapement is provided by the movement of the platen and the paper—you could consider this the "film plane"—and the image is always "generated" at the same position relative to the "font store" of striking keys.

On the other hand, we recalled that the IBM Selectric typewriter handles its problem of escapement quite differently. The platen does not move at all. It remains fixed (so far as the horizontal direction is concerned), but instead the typing element moves from left to right—again one-tenth or one-twelfth of an inch at a time.

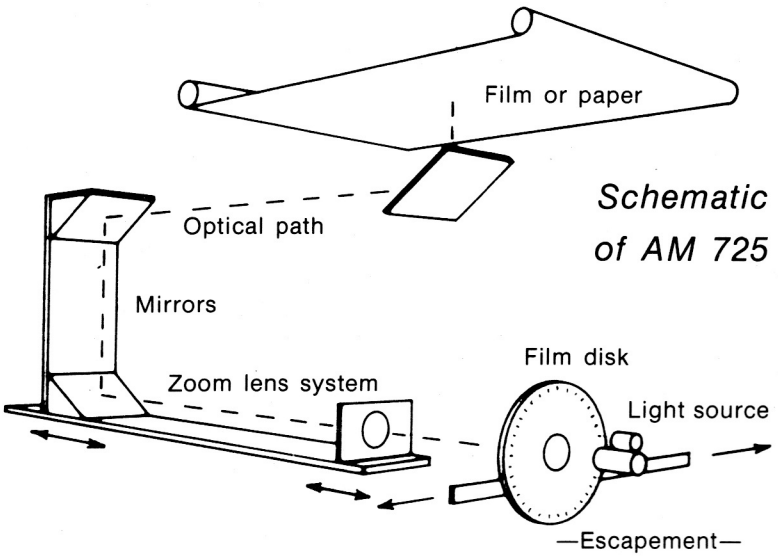
These are two different principles of "escapement" or "character laydown." In one case the "film plane" moves. In the other case, the "image carrier" moves. In both cases the movement is non-proportional—*i.e.*, in a monospaced fashion.

With second-generation phototypesetters the physical act of escapement may be handled in different ways. What are the alternatives?

1. The film carriage (comparable to the typewriter's platen) could move horizontally, and the image can thus always be projected from the same location. This is the way the Compugraphic ACM 9000 works, and the much older and more primitive ATF-B8. It is also the way the Fotosetter worked except that the carriage motion in that instance was vertical rather than horizontal.[1]

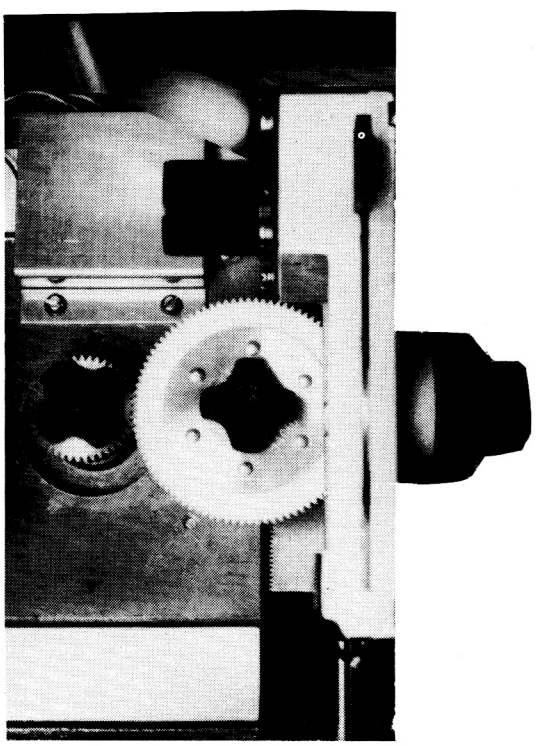
1. See illustration on page 74.

2. The image itself can move. Consider the AM 725 (shown at right in the diagram). The entire disk assembly mechanism, together with the light source, steps across the film plane, character by character, flashing each desired image, moving the necessary width, and then returning to the beginning for the next line as the film is advanced. The same principle has been used by the Bobst 132 and 142 typesetters, except that with this design, not only the film plane and flash, but even the lenses (which are very tiny and housed in a "pack") also "escape."



Schematic of AM 725

3. The image can remain stationary, and the film plane can also remain stationary, while something "in between" takes place to cause character images to position themselves appropriately.



a.) It could be the lens itself (as with Compugraphic's 4961) which steps across in front of the film plane, picking up the image of the flashed character and directing it onto the film plane directly opposite its current position. (*In the illustration at the lower right, we see one of two lenses in position, while the second—directly above the set gear—has been blocked off manually by the movement of a shutter. The lens not masked is stepped across the film plane by the paired set gears, also shown in the illustration.*)

b.) Or the lens can remain stationary, and the image as projected from the lens can be deflected incrementally by a prism, mirror, or similar device. This is the most common method of all, since it involves the least mechanical motion and requires moving a smaller amount of mass. And when we study the techniques used for displacing the image in this fashion, we will see that over the years the amount of movement, and the mass to be moved, has been constantly reduced by superior engineering.

Most of the displacement techniques are of the kind we have just described; where the image location at the time of exposure is always the same, where the lens position is fixed (except as it may be rotated or racked for sizing) and where the film plane is stationary (except as the film or paper may be advanced from line to line). Thus the projected image is displaced by means of a stepping mirror or prism, which deflects the picture of the character progressively further down the line of type, and then moves back and starts the "laydown" process over again on the next line.

In its simplest terms, a mirror is positioned at a 45° angle so that the image which is projected is deflected (usually by 90°) onto the film plane. Then the mirror, which is seated on a track and which is moved along the track by a set of matching gears driven by a stepping motor, backs up the required amount, and receives the next character flash. The gears which enable the mirror to step backward must be "teethed" in a manner fine enough to move by the distance as determined by the width of the character (assuming that the stepping motor "knows" what this distance should be).

For example, if the character were 4/18 of a six-point em in width, then the distance the mirror would have to move before (or after) exposing the character would be .018496". Or this value could be rounded off to the nearest ten-thousandth of an inch, such as .0185". In general, the stepping arrangements are such that while the "backward" movement of the character laydown is accomplished in fine increments, at the end of the line there is a very rapid return of the mirror or prism (the forward movement) so as to coincide with the film advance and to minimize the "dead time" before the next line may be typeset.

This moving prism or mirror technique was first employed by the Photon 200 and it was retained in the design of the subsequent Photon products until the Pacesetter line appeared in the early 1970's. Actually, the first application of the

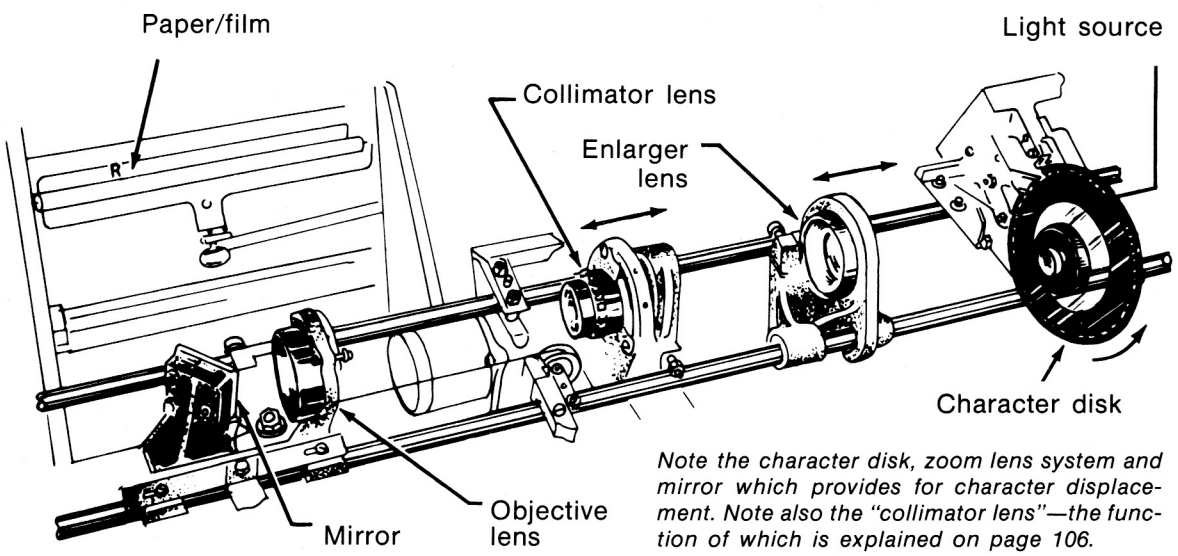
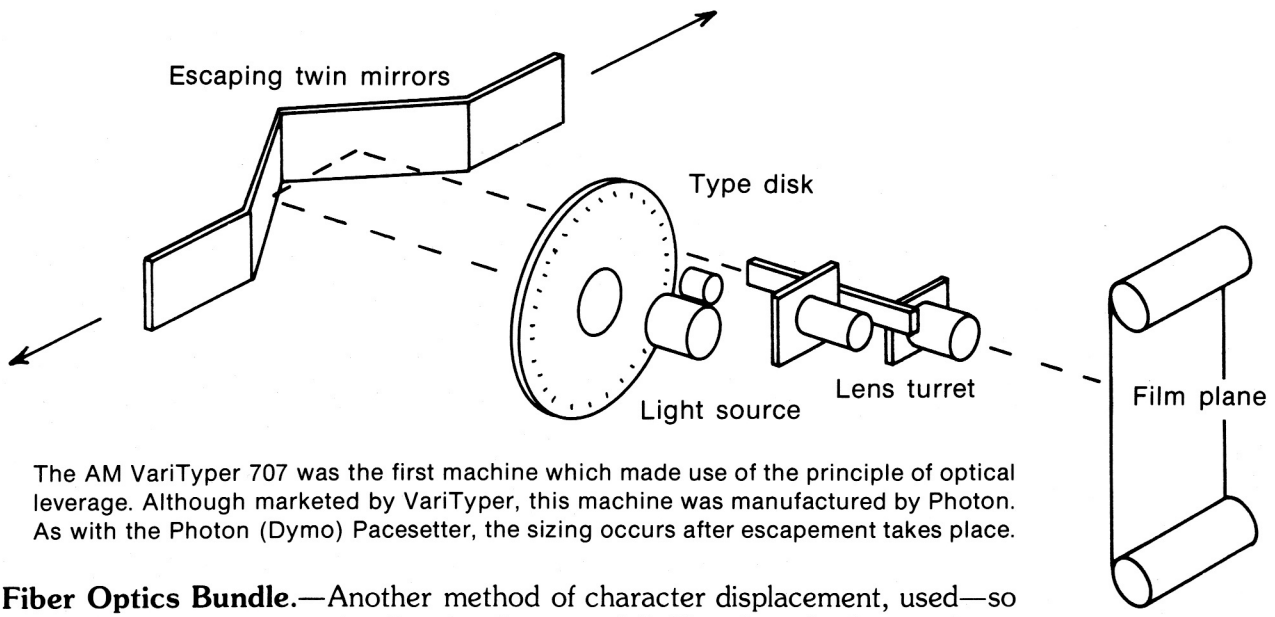


Diagram of the AM VariTyper Comp/Set 500

Note the character disk, zoom lens system and mirror which provides for character displacement. Note also the "collimator lens"—the function of which is explained on page 106.

new Pacesetter technique was developed by Photon for a machine which they designed and manufactured under a contract with Addressograph-Multigraph VariTyper in late 1971, called the AM 707. The concept was later applied to the 744 and 748 models. Here you will note that the escaping mirrors are, in fact, twin mirrors, and that the image is bounced against the first and then the second mirror before it is directed toward the film plane. This accomplishes what Photon called "optical leverage." In other words, the mirror moves only *one-half* of the required distance, but the effect is as if it had moved the entire distance because the angle at which the light bounces between mirrors changes constantly. As the mirror moves to the left, the spot of light strikes a point further from the center on the surface of the first mirror, is reflected to a point further from the center on the surface of the second mirror, and hence moves twice as far as the movement of the twin mirrors. Notice, too—and we shall return to this later—that in this case the image is *sized after the displacement occurs.*

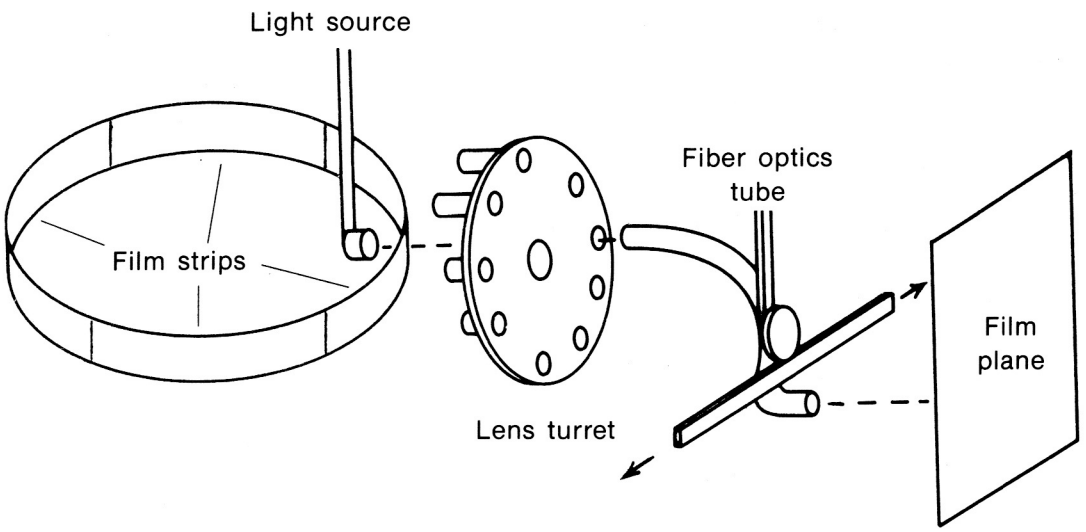
The use of optical leverage, in and of itself, reduces by exactly one-half the distance that the mirrors must travel. To set a 20-pica line, for example, a mirror movement of only ten picas would be required, and a "carriage return" of only this smaller distance would be necessary as well.



The AM VariTyper 707 was the first machine which made use of the principle of optical leverage. Although marketed by VariTyper, this machine was manufactured by Photon. As with the Photon (Dymo) Pacesetter, the sizing occurs after escapement takes place.

Fiber Optics Bundle.—Another method of character displacement, used—so far as we know—only on the Graphic Systems C/A/T and on the Singer phototypesetters which were manufactured by Graphic Systems, is to employ a fiber optics "bundle" to carry the sized image from the lens turret to the film plane. As can be seen from the diagram on the next page, the revolving drum consists of four segments, each of which can be mounted separately. A xenon flash "picks off" the character from the spinning drum and exposes it through the desired lens. However, there is no need for a straight-line optical path from this point on. The fiber optics bundle has the property of being able to conduct light—or even the characteristics of an image—up and down hill and around corners without the need to "bounce" or "direct" the light rays. The far end of the bundle contains the optical components necessary to recompose the image, and since

the bundle can be moved incrementally along a gear track, escapement can be determined by a stepping motor, and the image is thus always positioned exactly opposite the location at the film plane where the image is to be laid down. One limitation of this system appears to be that the bundle is not capable of carrying the elements of an image greater in size than 18 points, so, to increase the size repertoire from nine lenses so that 15 different point sizes are available, a second lens can be interposed between the end of the fiber optics tube and the film plane, doubling the sizes of the images transmitted through the turret and the fiber optics bundle.



Illustrating the fundamental optical and character laydown principles of the Graphic Systems C/A/T.

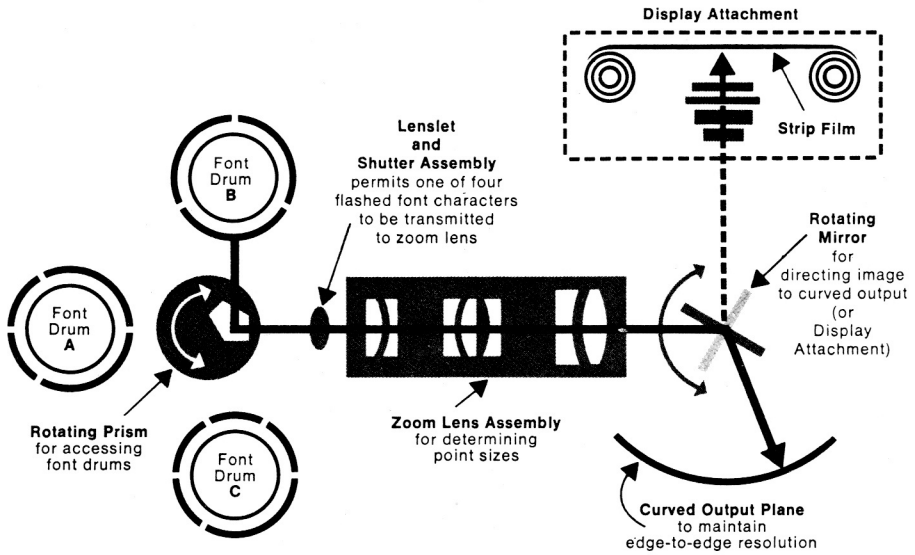
Certain other second-generation phototypesetting machines use fiber optic tubes to direct light to a particular location, but the Graphic Systems' design goes beyond this since it transmits the image itself. In a sense, what transpires at one end of the tube is a "scrambling" process, with the image being "unscrambled" at the output end.

The gear track is able to translate the movement of the tube into fine enough increments so that characters can be laid down, one at a time, with precision, at the appropriate escapement widths, considering the width of the master image multiplied by its point size. Fonts used by the Graphic Systems' phototypesetters are based on this principle, and the relative unit increment used is 36 to the em. Thus, unlike the Compugraphic 2900, 4900 and CompuWriter models, there is no need to change gears or pulleys according to the desired sizes.

The Rotating Mirror.—Another interesting method of character laydown involves no actual "stepping out" along a rack or track, but merely a change in the angle of a mirror which "sprays" characters across the setting line. Reference to the accompanying diagram illustrates the principle involved. The picture of the image is projected against the surface of a mirror and bounced off the mirror

through the lens system and onto the photographic film or paper. To achieve escapement nothing moves except for the mirror, which must change its angle (after the exposure of each character) by a very fine increment, on a character-by-character basis, in order to lay down each character in succession from left to right across the film plane. At least three typesetters have made use of this principle: the Harris Fototronic TxT (and the subsequent 4000 model), the Comp-Star 190 as well as other later models of the CompStar,[1] and, of course, the Mergenthaler V-I-P.

1. The Comp/Star, originally a product of Star Graphic Systems, is no longer offered for sale.

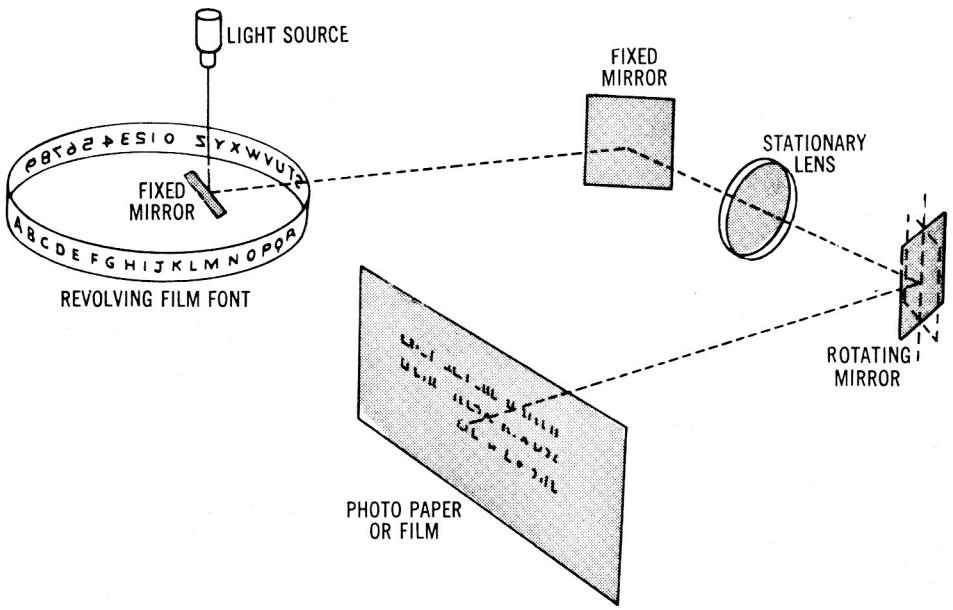


The V-I-P optical system

There is a serious problem of character or image distortion which must be dealt with when escapement is handled in this fashion. If you were to imagine holding a mirror, bouncing a light from a fixed source onto the mirror, and then rotating the mirror so that the spot of light would move across the surface of the wall (the film plane), you would notice two imaging problems: the first is that the focus of the image against the wall will change. The distance from the mirror to the ends of the hypothetical line of type will be greater than the distance from the mirror at the center of the line. Consequently, if your image is in focus at the center of the wall, it will be out of focus as you move away from the center. The second problem is that the shape of the image itself will tend to be elongated on the sides. If it is a circle in the center, it will tend to be an oblong on the outside edges of the arc of travel. *(See illustration on page 105.)*

To some extent the out-of-focus problem can be compensated for. How badly out of focus the image would be at the edges if it were in focus in the center of the line would depend upon the depth of field of the lens system—which is, of course, the distance that an image tends to remain in focus. Consider the manner in which the CompStar endeavored to solve this difficulty. As you can see from the diagram, the image was literally bounced around the cabinet in order to provide the greatest possible depth of field before it reached the displacing mirror.

Moreover, the Star design brought the line into the sharpest focus not in the center, but one-third and two-thirds of the distance across the line. Consequently, the image would tend to be barely out of focus (if at all) on the extreme ends of the maximum line length, and again, in the center of the line. But the image remained quite sharp across the entire line. On the other hand, there was a tendency for character shapes to become somewhat elongated horizontally as you moved away from the center.

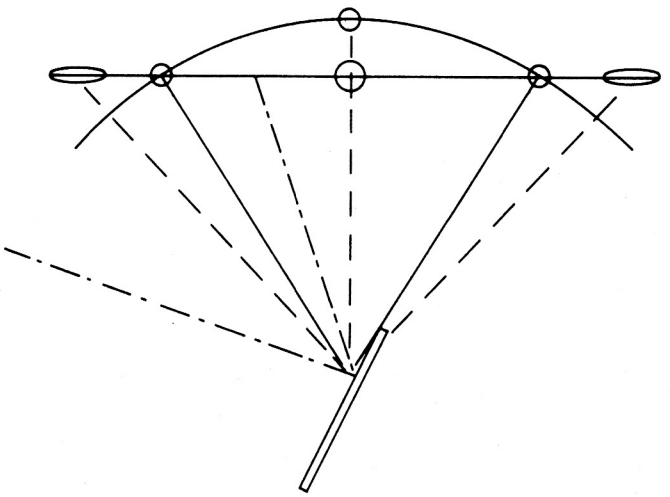


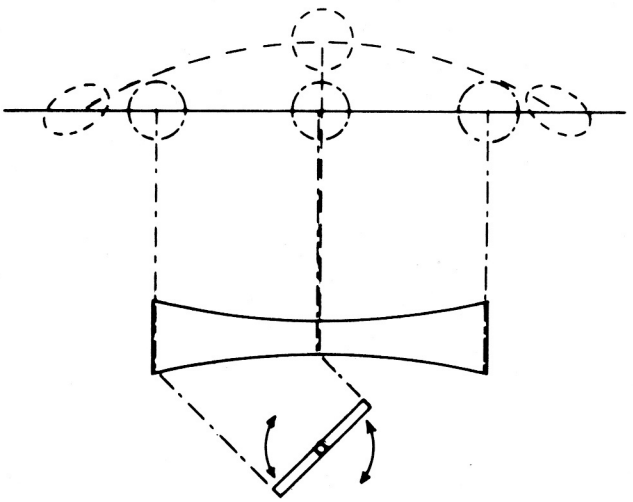
The CompStar typesetters increased the focal length to minimize distortion by bouncing light around the cabinet.

Harris resolved this problem by building a "field flattener" lens system, the function of which is to bend the light rays so that they remain in constant focus. Mergenthaler solved it in still a different manner, which is to form the film or paper into an arc, so that the distance from the mirror to any position on the arc is always precisely the same, no matter the position of the mirror. (*See schematics on the next page.*)

The mirror method of displacement has many advantanges: the mirror itself is tiny and can be rotated very quickly. There is less mass to move than even the optically-leveraged mirror system. And the mirror can return to the beginning of the line almost immediately, so that there is no delay in accomplishing the equivalent of "carriage return." On the other hand, the very, very infinitesimal change of the position of the mirror associated with the width of each character requires extremely precise control and a very fine stepping motor. The mirror must be able to move in such fashion that the next character may be only a few thousandths of an inch from the preceding character, and since distance is amplified by the travel of the image from mirror to film plane, the mirror movement must be only a fraction of the amount of escapement desired at the film plane.

The arc shows the plane at which a deflected image would be in focus. If the film (represented by the straight line intersecting the arc) is flat, the deflected image will be out of focus and distorted.





A field-flattener lens, such as that used by Harris on the Fototronic TxT and the Fototronic 4000, is designed to straighten the light rays, correcting for distortion and focus.

Escapement before or after sizing

If you will refer to the diagram of the "Photon" Pacesetter on page 107, you will note that there is a significant difference between the conception of this machine and that of many other phototypesetters. Compare it, for example, with the diagram of the Mergenthaler V-I-P or the CompStar 190—or, for that matter, with the original Photon 200. Note the escapement mechanism of the Pacesetter. It consists of a pair of mirrors which deflect the image so that it moves across the film plane, thus providing the character laydown. And, as with the AM VariTyper 707, the amount of mirror movement is only one-half of the escapement distance required, because of the principle of *optical leverage*. But study, particularly, the location of the sizing lenses in relation to the escapement mechanism. You will observe that the images are sized *after escapement*, whereas with most other typesetters, images are sized *before escapement*. (This distinction also applies for the AM 707, which, as we stated, provided in a sense the prototype for the Pacesetter, and which is illustrated on page 101.)

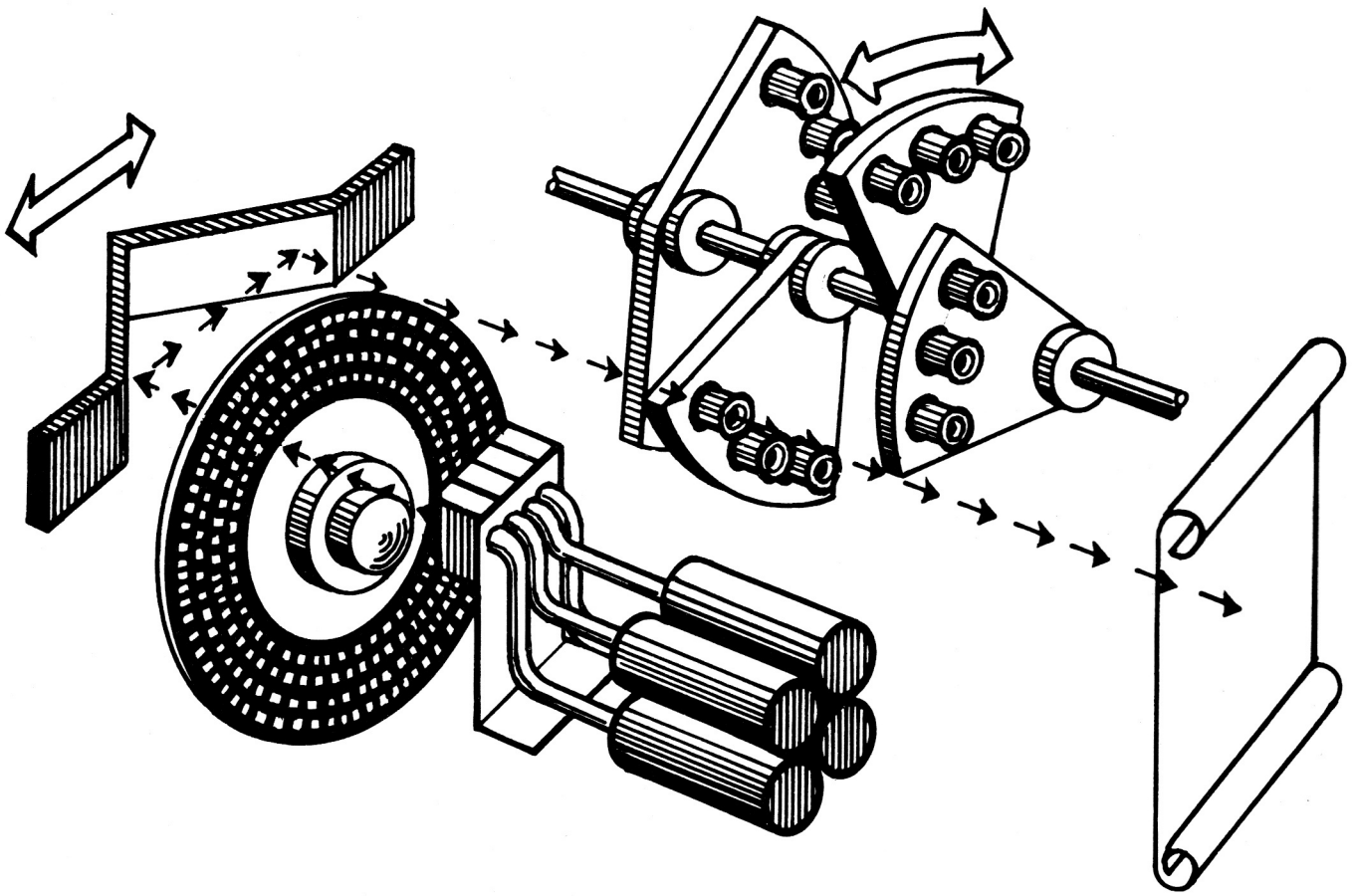
If characters are sized before they are "laid down," *then the escapement mechanism must, of course, take into account the actual physical size of the image as magnified or reduced at the film plane.* If they are "laid down" before they are sized, on the other hand, then the escapement mechanism need only consider the size of the master character as it is stored on the disk, drum or film strip.

The master images on the Pacesetter disk are five points in size (although they can be drawn from a 10-point or some other appropriate design). Due to the principle of optical leverage, in order to escape by the width of one five-point em it is only necessary for the mirror system to move two and one-half points, even though you are setting a ten-point character—or, more specifically, a character the width of a ten-point em. Normally, the escapement would have to be ten points, but optical leverage reduces this to five. However, since magnification takes place *after* escapement, the mirror system moves only by two and one-half points, thereby reducing the motion by one-fourth that which would ordinarily be required. And suppose you wished to set a character the width of a 72-point em. Again, the escapement, instead of being 72 points, *would simply be two and one-half points.* Thus, to set an entire 45-pica line of 72-point characters, you would only need to lay down a 7.5-pica line of five-point characters. The mirror movement, instead of being 45 picas in each direction, would only be 18¾ points, or 1.57 picas. Figure this out for yourself: In a 45-pica line there are 540 points. You can fit only 7.5 72-point ems into 540 points. And since you are actually only setting five-point ems, you will require 7.5 ems of five points each, or 37½ points. But because optical leverage reduces this by half, the mirror movement would come to only 18¾ points! As you can see, the combination of these two principles —sizing after escapement and optical leverage—significantly reduce the amount of motion or "travel" required.

There is another advantage to sizing after escapement, and that is that you do not have to calculate the amount of escapement according to point size. If you size *before* you escape then you must multiply your unit width values times the point size. It is not necessary to do so if sizing is not a factor in the escapement calculation.

If you size after escapement, in all cases the machine escapes the same amount regardless of point size: namely 36 units per five-point em. But due to optical leverage the escapement value of each five-point character is reduced by one-half. The lens system takes care of all sizing problems. However, there are certain drawbacks to the sizing of type after escapement. Probably the most important is that it is difficult or impossible to set to the maximum measure in all point sizes. If you are able to set a 28-pica measure in five points, you will be able to set 33 picas in six points, not more than 39 in seven points, 40 in eight points. At nine points you can go to 45 picas. The line length at 10 points is 50 picas, and at 11 or above you can go to the full width of the Pacesetter film carriage, which is 54 picas. Moreover, since the enlargement takes place in a horizontal direction as well as vertically, it is not an easy task—and sometimes it is impossible—to maintain a constant left-margin alignment of text. Some special "fiddling" must be

accomplished to obtain alignment of six-point type at the left margin, for example, and then only at the expense of the maximum width you can set in the same take for larger point sizes. This limitation has not been shown to constitute a substantial hardship, however.



The basic principles of the Photon (Dymo) Pacesetter

There are four xenon flash light sources, and each is directed to a specific row by means of a fiber optics bundle or light tube. The character, once exposed, is bounced against twin mirrors which minimize the escapement distance according to the principle of optical leverage. Note also the interesting feature that the escapement takes place before the images are sized. The lens turret shown here could offer a total of 16 sizes, but focusing and sizing are accomplished in part by locating the lenses at different distances from the mirror assembly. This allows, among other things, for the larger field of vision required by this system.

For the most part, second-generation typesetters size before they escape, and so the amount of escapement must be calculated in relation to the desired point size. When you come right down to it, escapement values must always be executed in *absolute* terms, because whatever it is that moves to provide the escapement must be expressed in some increment of a fraction of an inch or centimeter. Escapement values are derived by multiplying the unit value of the character by the point size in question. In other words, with an 18-unit system, you escape twice as far for a ten-point character as you would for a five-point character, and if you have a five-and-one-half-point lens, the computational values

are even more complex. Almost invariably there is a rounding error, even when the ultimate escapement is reduced to a displacement as fine as one ten-thousandth of an inch. The electronic logic (if there is any) must not lose sight of this rounding error since when the totals are cumulated and they add up to too much or too little, some compensating adjustment must take place. The alternative is to vary the measure from the "norm" according to the point size being set, and even this technique will not work if you mix different point sizes in the same line.

But how does the typesetter "know" in the first place how much to escape on a character-by-character basis? This brings us to the next topic, which has to do with the technique for imparting to the typesetter a "knowledge" of width values.

Escapement values

In the beginning of this chapter, we indicated that we would discuss the principle of escapement as one of the major concepts relevant to an understanding of the photocomposition process. We have already discussed the physical means by which the escapement process takes place—the movement of platen, film store, character store, lens, or some other kind of "laydown" device. But the other half of the problem is to figure out a way to tell the typesetter what the character widths are that must determine the computation of the escapement values. Then (perhaps) these will be converted from relative to absolute values. Here are some of the typical solutions to the problem of relaying width value information to the typesetter:

1. Pre-determined "hard-wired" values.—There have been a few machines for which the escapement was restricted to a minimum number of alternatives, and these were hard-wired into the machine. We have in mind the so-called "Justowriter" font machines, such as the Justotext 70 and 71 models and the Compugraphic 2970. Character widths for these machines varied only from two to six units, much like an IBM Executive typewriter. A lower case "a" would always have the same value, regardless of the font. It would be a relative value, depending upon the point size, but whether the "a" is bold, roman or italic, serif or sans serif, pseudo-Times Roman or pseudo-Helvetica, the width of that "a" was always four units.

Unit count fonts.—There are a few more machines which use only unit-count or unit-cut fonts. Unit-count widths are those used in the justification of newspaper wire-service tapes. Since all fonts have the same widths (even though there are finer increments and more variations between characters than is the case with "Justowriter" fonts), the font width tables can be wired into the "logic" of the typesetter. There need be no "table look-up" and no "plug board." The Execu-Writer and the CompuWriter Jr. (both products of Compugraphic) fall into this category. This was also true of the original CompStar 190. Despite the homogeneity of width values for given characters in all type faces, some of these faces nevertheless appeared quite varied and attractive.

2. Unitized fonts.—Other typesetters use true "unitized" fonts, and by this we mean that every type face may be designed differently from every other (although some faces may in fact be "duplexed" or "triplexed" so as to simplify the storage of width values). These type faces will be designed to a relative unit system, and the unit count is most likely to be either 18 or 36 units to the em, although there are some machines that use a different common denominator. For example, the CompStar line claimed to use a common denominator of 1/1000 of an inch, although in point of fact their width tables, being expressed in an eight-level code, allowed for only 256 different character width alternatives, and, in actual fact, their formula expressed individual character widths as 1/11 of 1/18 of an em times the point size, rounded out to the nearest 1/256 of a point, or .000054″. The Harris second-generation typesetters have used an absolute value which they term a "piclet" and which is equal to one-tenth of a point, or 1/720″. Subsequently, Harris introduced the "triclet" which divides "piclets" into thirds. But the critical point is not so much how the type is designed (and to what base)—although of course this is relevant—but rather how the width values are stored or described in computer language or electronic logic.

For example, suppose you devised a system which allowed only four combinations of "bits[1]" to describe character widths. It would thus seem that such widths could be expressed as variables ranging from 0 through 15, since four bits offers only this many combinations. On the other hand, if you determined that no character or space could be smaller than 3/18 of an em, you could simply add three units to the top of the expressed values and obtain a scale ranging from 3 to 18 and you could, in all fairness, call this an "18-unit system" since one character's width could vary from another's by as little as 1/18 of an em, providing that no character's width fell below 3/18. In like manner, using a five-bit method of storing character widths, which would give you 0 to 31 different possibilities, and adding six units, you could express character differences in increments as small as one out of thirty-six within a range of 6 to 36.

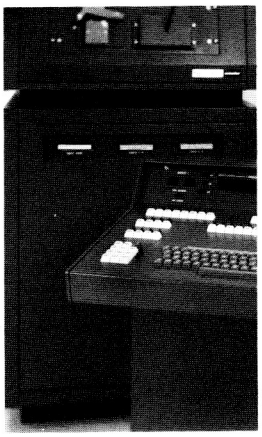
1. See the discussion of "bits" on page 167.

One must ask, then, 1/36 of what? No one, presumably, is going to set type in a one-point size, and yet it still may be convenient, inside the typesetter, to work in absolute units, such as points, rather than relative units (particularly if you want to mix different point sizes in the same line) and consequently you may go through the arithmetic calculation of multiplying all relative width values by the point size in order to come up with a common absolute denominator the lowest theoretical value of which is 1/18, or 1/36 of 1/72 of an inch.

On the other hand, there are some typesetters which offer the ability to set type in half-point increments—especially for the 5½ and 8½ point sizes. And there are also typesetters which offer the opportunity to "size" type in one-tenth-point increments, which would suggest that the finest increment in which a type character's width might be expressed could be 1/100 of 1/10 of 1/72 of an inch!

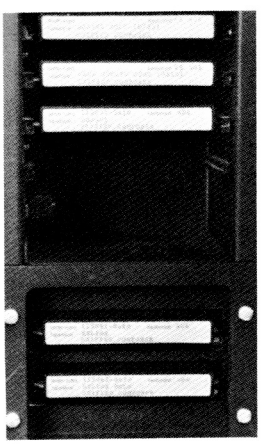
Width value storage.—But for our purposes the most relevant point is how the widths—whatever they are—can be conveniently *stored* within the typesetter, *or fed* to the typesetter, so that it can take this information and make whatever

additional calculations it may need in order to figure out how far to move a prism, to rotate a mirror, to shift a carriage, to advance a fiber optics tube, or to move the film plane itself. And again we return to the fact that the machine must either use predetermined width values, or else it must have some means of ascertaining them. One approach is for the typesetter to be *told* what each width value is when it encounters the code calling for a particular character. This may be done in one of two ways: the machine can read the width value from the font disk itself. Or, it can be provided with the character width information along with the character code on the input stream. For certain "third-generation" (CRT) typesetting machines, as we will discuss in the next chapter, the character width may be derived from the stroking pattern which makes up the character itself. (More about this later.)



Compugraphic uses type font width cards in its UniSetter, as may be seen in the illustration above.

Below: Width cards are also stored in Compugraphic's Unified Composer, where input to the typesetter is produced.



Except for those second-generation machines using only Justowriter or unit-count fonts, somehow the typesetter must come into the possession of character width information. One method of "telling" the typesetter what width values to use is to provide for the insertion of a "plug board" or "plug boards" into the machine, connecting with its electronic circuitry. When the code for a certain character and font is encountered an electrical circuit is directed to a certain location on the plug board where it picks up, for example, a certain number of pulses which would cause a stepping motor to rotate a certain number of revolutions. The electronics often have to be more complicated, of course, because it may also be necessary to know the desired point size of the font, so that the basic width value can be multiplied by point size.

In the past, plug boards have been cumbersome and relatively expensive. Nevertheless they were widely used in the "early days" in order to describe width values. Even more recently, the Compugraphic EditWriter requires the insertion of such plugs. The input program or counting technique, of course, had to produce results which agreed with those stored values. With Linofilm, on the other hand, width values were output on the tape along with each character code, and the operator, at the keyboard, had the option of varying that value in order to achieve certain kerning relationships.

Alternate methods of storing width values

Starting with the Photon 713, however, it became quite common to depart from the use of the width plug to "load" width values into a "memory" within the typesetter itself. These typesetters had their own memory—at least large enough to store all width values required for a given font "dressing." Hence, when a code was encountered which called for a certain font and character position, the width of that character could be ascertained from a "table look-up" and the escapement calculated accordingly. When the machine was "dressed" with the desired fonts, widths were loaded at the same time, usually from paper tape. A diagnostic panel was provided to assure the operator that the values were "read in" properly. (In fact, with the Photon 713 you could also load in a table which would identify the code configuration to be used to describe each machine function and to identify each character position.)

Storing width values on the film disk.—As we have indicated, character widths can also be stored on the type disk or film strip itself. This was first done on the Fototronic 1200. Photodiodes were used to decipher various combinations of bars and "non-bars" (transparent areas) in order to ascertain from these configurations what the individual character widths were. Later, an alternate approach was developed by certain manufacturers whereby all width values are "read" from the disk at one time and stored into memory when the machine is "primed." This is the same technique employed by the Alphatype AlphaComp, the VariTyper Comp/Set series and Comp/Edit, and the Compugraphic VideoSetter.

The "slave" typesetter.—When we use the term "slave" typesetter we mean, among other things, that the typesetter has no way of "knowing" the width values it must consider, except on a character-by-character basis as a part of the text input stream. Most typesetters, however, are not "slaves" in this sense. They have enough "intelligence" to know what width values they are dealing with. But the original Linofilm output device read a 15-channel tape which conveyed character width information as well as character and font identification. Later a three-frame six-level tape provided the same data. The Photon 530, 540 and 560 models also required a multi-frame tape for the same purpose, as did the Graphic Systems C/A/T.

Once the width values are known to the machine, then the mechanics of escapement can be accomplished. This will involve moving a mirror, advancing a prism along a rack, or otherwise displacing the image onto the film plane.

The justification function

Most of the early typesetters also had to be told the value of the interword spaces which had to be inserted in order to make lines justify. Even if they knew the width values of characters and could lay down such characters with the necessary amount of escapement, they did not necessarily know the value assigned to interword spaces. Hence, at the keyboard level, computations were made which provided this information. In this connection several possibilities were available to the designer of phototypesetting devices:

- A counting keyboard could provide precise information as to the value of each interword space.

- A counting keyboard could provide information as to the total value of all interword spaces (or the excess over an established minimum value), and perhaps also the number of interword spaces, so that the typesetting device could make its own calculations.

- The keyboard could merely provide information as to the point at which the line ending would occur, and expect the typesetter to add up all width values, count the number of interword spaces, ascertain the "deficit" that had to be made up, and distribute this deficit among the available interword spaces

(or perhaps even between characters if letterspacing were necessary and permitted).

• There are some phototypesetters which cannot make their own calculations. These "slave" typesetters have to be told by the input keyboard (or computer system) just how much space to allow for each spaceband. But quite early in the history of the development of photocomposition it became obvious that phototypesetters should be able to accept and to process TTS tape, just as it was produced to drive linecasting machines. And linecasters could *justify* lines that had already been specified as being within *justification range,* by virtue of the mechanical feature of spaceband expansion. Thus it was deemed necessary to design phototypesetting devices which, somehow, could accomplish the same results—that is, to compute the value required to be inserted between words in order to cause a line to justify after it had been brought within justification range. This meant, of course, that some justifying *logic* had to be built into the typesetter. When we speak of justifying logic in this sense we do not mean logic which counts character widths *in order to make end-of-line decisions.* We merely mean logic *to implement* end-of-line decisions *that have already been made* by the keyboard operator. As we shall see, phototypesetters nowadays frequently contain another kind of justification logic, which permits them to make "their own" end-of-line decisions.

In order for the typesetter to compute the values to assign to interword spaces when statistics for this purpose are not provided in the input stream, the typesetter must look up the character widths and decrement them from a counter which establishes overall line length. It must also count the number of spacebands (variable spaces) within a line, and ascertain the "deficit" to be divided among the spacebands. We shall discuss this function further in our consideration of the justification process in Chapter 12. But it should be evident that some fairly sophisticated and very rapid arithmetic calculations are required before each line of type can be typeset. The entire line must be processed before the first spaceband code is "escaped" and hence before the first character in the line can be typeset. In some cases this was accomplished by providing the typesetter with two paper tape readers. One read the tape until it encountered a line-ending code reporting its findings to the "logic" unit. This enabled the machine to make its spaceband calculations and set the interword spacing widths accordingly. The tape was then passed to a second reader which processed each character for exposure.

A second approach was to read the paper tape until an end-of-line signal was encountered, set the spaceband widths, and then *back up* the tape to the previous end-of-line code, and re-read the tape character-by-character for processing through the actual character laydown and setting procedure.

The most common method today, however, is to read the line into a *buffer* which stores the codes for the line, and then to examine the information in that buffer first to make the necessary interword space calculations and subsequently to set type. In the meantime a new line could be read into that or an alternate buffer.

Initially these interword space calculations were made by what we call "hard-wired" logic. More recently, most phototypesetters of the "second-generation" variety contain mini- or microcomputers. These computers serve as typesetting "controllers" and also perform other functions. As typesetting controllers they keep track of character positions on the revolving disk or drum, they trigger the xenon flash to expose characters at the appropriate time, they convey information on how far to escape after exposing the character, and they inform the film advance mechanism when and by how much to move the photographic film or paper in order to compose the next line.

As we shall see in the next section of this chapter, these small computers can perform other functions as well. They can actually make the end-of-line decisions, and even determine where and how to break words in order to achieve better justification. They can also simplify the keyboarding and typesetting process by storing "formats" or character strings which may be used frequently and thus achieve quite complex typesetting results, such as arranging material into tabular formats.

Comparing typesetter characteristics

In order to provide the reader with information for understanding and evaluating second-generation phototypesetters, we have listed the headings which appear in the chart which we publish annually in "The Seybold Report." We have not reproduced the charts themselves since they quickly become obsolete. But the reader may wish to gather his own current information from vendors and make up his own current chart. This would also be a good classroom project. To help you, we include, as a supplement, a current chart which he have prepared.

Machine and price.—In the chart or table, the price given should be the base selling price f.o.b. the factory and intended for the American market. The price generally includes set-up and installation in the plant, one set of film cassettes, and one standard "test" font supplied by the manufacturer. To this must be added the cost of shipping the machine, the cost of fonts, the cost of extra film cassettes and spare parts, and the cost of additional lenses that may be purchased, if lenses are interchangeable. There are other options, and an indication should be provided of what they are and how much they cost where the information is readily available and can be conveniently summarized. Some of these options have to do with additional computer memory and front-end software.

Speed.—Here you would state the manufacturer's claimed speed for his equipment when setting "straight matter." In all cases the output speed of the phototypesetter is specified in terms of the number of nine-point, eleven-pica newspaper lines the machine can set in one minute. (This is designated as "lpm.") It may not be assumed, however, that the relative speeds reported would be relevant outside of the newspaper straight matter context. Even for straight matter, some machines lay down characters faster than others, but "lead" slower, or *vice versa*. Consequently, a linear relationship does not exist for wider measures. Moreover, font changes, size changes, and additional film advance (leading) will

take much more time on some machines than on others, and will usually require some additional time on *all* machines. Two machines with the same claimed output speed on straight newspaper copy may perform quite differently on more complicated material. The only way to gauge throughput accurately is to take copy considered to be typical of the kind of work the user expects to do and to set it and time it on the machines he may be considering, bearing in mind that he may be able to optimize speed by customizing the font and lens arrangements, or even, in some cases, combining different sizes on the same film carrier.

Some machines obtain higher speeds by reducing the number of characters available, in order to provide character redundancy and to flash more characters during one or more revolutions of the spinning disk or drum. When it is possible to run these machines in two modes of operation (one with fewer characters and higher speeds), both speed ratings and both sets of character complements should be included.

Flexibility.—The versatility of the typesetter (apart from its speed) is a function of the characteristics set forth in the paragraphs below, represented by italicized lead-in topics.

Number of characters. This is the number of individual character image masters available on the machine at one time. On most machines (but not all, by any means) there is at least some time loss when you mix or select characters from different fonts. And, as stated above, the number of characters may be reduced if a higher running speed option is elected.

Number of sizes. This is the number of lens positions or stops. Sizing may be achieved by rotating a different lens into position, or by moving a "zoom" lens arrangement. In general, the lens rotation solution involves somewhat less time loss.

Some machines allow the user to mount larger or smaller sized font masters and thus to produce a greater range of point sizes from the same lens complement. This, however, may tend to reduce the number of different character masters available on the machine (as in the case of the "B" fonts on the Mergenthaler V-I-P). Therefore only the number of sizes it is possible to produce from one character master should be listed, and not the total number of sizes possible with different sizes of font masters in combinations with different lenses or lens stops. If manual intervention is required to change lens size to get a size other than that provided by a given master/lens relationship, the number of sizes is listed in parentheses, thus indicating that not all of these combinations can be accessed "under tape control."

Number of characters times number of sizes. This is a calculation obtained by multiplying the total number of individual characters available on the machine by the number of sizes in which each character may be produced. This calculation is useful because it provides at least a rough index of typographic flexibility. If manual intervention is required to change lens sizes, the "character x sizes"

figure could be listed in parentheses. (Some machines permit the user to substitute lenses and thus obtain different combinations of sizes and fonts.)

Size range. This heading could be misleading if it were to imply that the machine is in fact capable of providing all of the sizes within the stated range, as from 5-72 points. In fact, only *some* sizes within this range are usually possible, at least at one time, and the reader will need to refer to the *Number of Sizes* summary to find how many sizes one can get. A chart might also indicate whether or not there is any flexibility in the selection of specific sizes within the range. In other words, if you can have 15 sizes between 5 points and 72, which 15? In some cases certain of the sizes available may not be very useful.

Fonts.—The heading *Number of font masters* would refer to the number of replaceable font elements (disks, grids, film strips, *etc.*) which can be mounted on the machine at one time. The higher the number of font masters (and hence the lower the number of fonts contained on each master) the greater the flexibility the user has in constructing the proper font complement for any given job as he moves from job to job. At one extreme there are machines such as the Pacesetter which use only one font master containing four to 16 fonts at one time.[1] At the other extreme, typesetters such as the V-I-P allow a separate font master for each font to be used. The user may thus make any combination of fonts he wishes out of the library of faces he has purchased.

Price of font masters is the cost of a standard font (disk, grid, film strip or whatever)[2] Generally speaking, manufacturers prefer to sell one of several standard character arrangements. Special or "custom" character complements generally cost appreciably more. Specially licensed fonts, such as those obtained from the International Type Face Corporation, may also cost more. Bear in mind that the price of a font master is not the same thing as the price per font. A font master may include 16 fonts, or only one. (In some instances, the font master is only one or two characters and not a font at all!)

To arrive at an appraisal of the cost of fonts it is necessary to consider whether redundant fonts are required. For example, if working from non-segmented disks, Times Roman might be required on three or more disks, perhaps in combination with Univers, Helvetica, or Souvenir.

Machine characteristics.—This is a catch-all heading for a number of mechanical constraints which are built into the typesetter. Usually these are "hardware" limitations, having to do with the way in which the machine is built—for example, in how fine an increment the escapement mechanism can move. But they can also be software-related, having to do with code translators and the like.

Maximum line length is, quite obviously, the widest measure (in picas) which the machine is capable of setting. There are, however, a few machines which cannot set their maximum line length in sizes below, say, nine points. Such limitations could be set forth for each machine. The buyer should also consider that for such machines, if he wishes to mix sizes, he may not obtain left-hand

1. Two fonts are arranged in one circle on the disk. There can be eight such rows or circles of type. However, the high-speed models replicate fonts to increase the speed of setting. To obtain the same character it is not then necessary to wait for a complete revolution of the disk. Dymo now offers *segmented* disks which provide greater typographic flexibility.

2. In some cases you must also pay for width cards and/or gears associated with particular fonts.

margin alignment, at least for maximum measure setting, for, say, twelve and eight-point type.

Maximum take length is the maximum length of photo material which can be exposed before the operator must remove the material from and/or load new material into the machine. Most of the typesetters use roll film or paper and in these cases the length of the maximum take is dictated by the capacity of the output or take-up cassette, which is generally smaller than that of the supply box or cassette. Some machines use only sheet film which is, of course, limited in size.

Input tape format refers to the format of the input (paper) tape which the phototypesetter will accept. Unless otherwise specified, it is assumed that the machine reads punched paper tape. (This does not mean that the machine must be equipped with a paper tape reader, however, since on-line use from a computer or video terminal, for example, is increasingly common.) Moreover, many manufacturers offer input from magnetic cassettes or industry-compatible magnetic tape. Paper tape is less common than it used to be, so the heading "under tape control" which appears below really means functions that the machine can perform from external codes which are read or received from the typesetter.

Some machines will accept only 6-level TTS-coded tape. Others can be "programmed" to read different input code structures. Still others can be programmed to read 7- or 8-level paper tape in different code configurations as well as 6-level input. Slave machines may require both the character code and character width information on the input tape, and some external device (keyboard or computer) must perform the justification or spacing out for the typesetter. Where width values are so conveyed from an external source the input is usually "multi-frame" in character.

Font width refers to the width-counting scheme used for character widths and for justification. The crudest designs for proportionally-spaced type were the Justowriter fonts. The next step up is represented by "unit count" fonts. The majority of typesetters use "unitized fonts" and in this case the number of units per em should be shown as an indication of better quality type design. However, it does not follow—although it should—that the more units per em the finer the quality of the type.

Finest leading increment should indicate the smallest amount of "film advance" you can command. Most machines provide leading in one-half point increments, thus enabling you to set, say, ten on 11½ leading. But for spacing out purposes, 1/10 point is extremely useful, especially for full-page make-up.

Reverse leading is a very useful feature. It indicates whether the machine can back up the film or paper to permit the setting of a second-column alongside the first, or otherwise to perform more complex setting and perhaps facilitate page make-up.

Functions under tape control.—These are the functions that can be performed by remote commands embedded in the input stream without operator intervention at the typesetting machine level. If the functions could not be performed under tape control it would be necessary for the operator to reset certain dials or switches or even to open up the machine and make changes internally. Change in *line length,* change in *point* size, and change in *leading* are such functions, as well, of course, as change in *type face*, which may usually be assumed.

Wire service input is a useful heading, with subheads *Accepts wire service tapes* and *Strips wire service tapes* to relate to the use of the machine to set from news wire service input, either observing the end-of-line decisions the input may already contain, or stripping out these end-of-line decisions and making new ones.

Justification.—This heading would relate to the ability of the typesetter to accept tape which does not already contain end-of-line decisions. *Hyphenless justification* would be one possibility. This would mean that the machine can break lines but will only do so without trying to break or hyphenate words. *Logic hyphenation* means that the front-end intelligence or computer/controller in the typesetter has been programmed to hyphenate at least some words by the use of a logic routine. *Exception word dictionary* means that this routine will also take advantage of a store of at least some words which cannot be hyphenated successfully by means of the typesetter's logic routine.

A subheading *Indents* could refer to the ability of the front-end intelligence of the typesetter to accept command codes embedded in the input stream which call for automatic indentions, and to execute these without operator intervention. Three standard types of indents are usually provided: *left, center*, and a left *"hanging"* indent.

Front-end intelligence.—This heading would indicate the growing importance of the "logic" or "intelligence" which is wired or programmed into the front end of most of the newer typesetters. Not only do most machines include some sort of hyphenation logic, but many have logic for tabular work and for stored formats as well.

An indication of the extent of the front-end intelligence can be gained by finding out as much as possible about the *front-end computer* that is built into the machine. In some cases the logic is "hardwired" or is "firmware" and is not readily susceptible to change, even by the supplier. In most cases the device offers a computer which may be reprogrammed. The size of such a computer in terms of "bytes" should be ascertained, although size is not necessarily an indication of efficiency or power. One use of the front-end intelligence of these computers or logic units may be to provide *automatic white space reduction* for larger point sizes, so that they will fit better, *tabs* for tabular setting, and *stored formats,* the power of which may be judged by the size of the *format storage area.* These are relevant subheadings for your tabular analysis of machine characteristics.

Review Questions:

1. Are second-generation phototypesetter character images stored as film positives or negatives?
2. Is the stored master image the true point size for which the character was originally designed?
3. How are characters "selected" during the composition process? How does the method of selection relate to the manner in which they are stored—whether on a grid or a disk, for example?
4. Since most phototypesetters can set images of different sizes, how does the sizing occur?
5. Can you think of a good reason for changing the master image for each point size?
6. What is the difference between a "zoom" lens and a "turret" lens system?
7. Why are phototypesetters concerned with the problem of escapement? What different methods of escapement are available and what are the relative merits of these different solutions?
8. How has "fiber optics" helped in the design of certain phototypesetters?
9. What is "optical leverage"?
10. What difference does it make whether the image is "sized" before or after "escapement"?
11. How does the typesetter "know" how much to escape for a given character?
12. Does the storage of width values present the same problems for Justowriter fonts, unit-count fonts and unitized fonts? What design differences might have to be considered?
13. If line-ending decisions are already present in the input tape, why does a phototypesetter have to be concerned about the justification process?
14. What factors influence the flexibility or versatility of a second-generation phototypesetter?
15. What design factors would affect the speed of the phototypesetter when setting straight matter?
16. Why do some phototypesetters handle "mixing" of fonts or sizes more effectively (i.e., faster) than others?
17. Why do some phototypesetters succeed in changing sizes faster than others?
18. What are the arguments in favor of having fonts arranged as film strips to be mounted on the machine separately rather than all on the same type disk? What are the arguments for having it the other way?
19. What is "front-end intelligence" in a phototypesetter and what functions can it perform?
20. Why have we sometimes used the term "computer" and sometimes "controller" when referring to the "front end" of the typesetter?
21. As an exercise, select a sample book or other example of typeset material and try to determine which typesetters could or could not handle that type of work "under tape control."

9

Third Generation (CRT) Typesetters And Subsequent Technologies

THE CONCEPT of applying "generations" to typesetters was probably borrowed from the computer industry, and in that instance it is presumed that each "generation" introduces an entirely new technology. The first generation of *computers* made use of vacuum tubes. The second generation introduced transistors and "solid state" concepts. The third generation involves the use of monolithic integrated circuitry and semi-conductors. In like fashion, the first generation of phototypesetters suggests the adaptation of hot metal (or impact) devices to set type photographically. The second generation implies that entirely new machines were especially conceived and designed to set type by photomechanical means. A *third-generation typesetter is one which does not in fact expose type directly from photographic masters but reproduces them electronically on the face of a cathode ray tube* (like a TV screen). The images thus created are photographed (either same-size, or enlarged or reduced by means of a lens system) directly from the face of the screen. If this seems a roundabout way of setting type, there are nevertheless advantages. One of them is that the image on the screen can be manipulated electronically in a variety of interesting ways. Another is the speed of setting. A third is the absence of mechanical parts and motions which present maintenance and reliability problems.

In the final pages of this chapter we shall also consider whether the techniques we describe here may not be outdated by even newer and more radical approaches to typesetting. But we will devote most of this presentation to third generation devices.

We shall learn that there are two basic categories of CRT typesetters. One creates a character image on the face of the cathode ray tube by simultaneously scanning a photographic master. Hence the "character store" is photographic, even if the image is not exposed on the film by the conventional means of shining a light through it. What happens is that the information derived from scanning this photographic master is converted into analog signals which in turn instruct the deflection circuitry of the cathode ray tube, telling it where and how to draw or paint the output character.

The alternative approach is not to store the character as a photographic image but rather as a digital representation. It is true, of course, that most characters are scanned in order to be converted into a digital form. But the scanning process is a different operation which takes place only once, when the information is initially created for subsequent storage in the typesetter.

Both of these basic types of CRT typesetters—those with photographically-stored character masters and those with digitally-stored character masters—will meet the requirements of the following definition of a "third generation" typesetter as one which:

a.) generates characters on the face of a CRT as strokes or dot patterns;

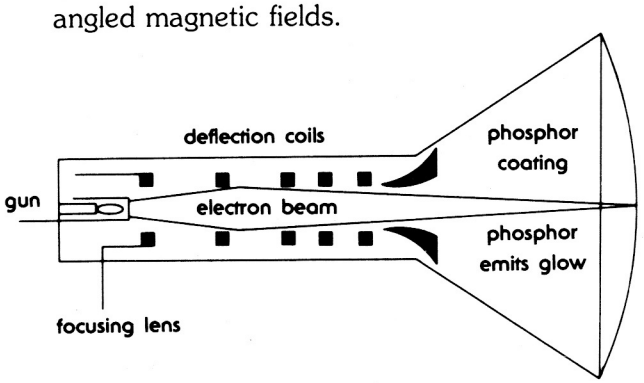
b.) accepts input primarily from a computer rather than directly from a keyboard (although some CRT typesetters have "intelligent front ends");

c.) operates at high speeds—usually substantially in excess of several hundred characters per second;

d.) and is frequently used to output fully-composed *pages* of type (as opposed to galleys), or is purchased with the expectation that sooner or later the device will be so used.

What is a cathode ray tube?

The illustration on this page indicates the manner in which the cathode ray tube works. Such a tube contains a cathode which emits a stream of negatively-charged particles of electrons. These electrons are subjected to a high-voltage acceleration in an electron gun and this action creates a beam of charged particles which is capable of being concentrated or deflected, either by electrostatics or magnetism. A grid pulse controls the emission of electrons. Anodes within the tube focus the beam. Deflection is caused by changes in the potential across horizontal and vertical electrostatic plates or by deflection coils that create right-angled magnetic fields.



Functional diagram of a cathode ray tube as used for character generation in a CRT typesetter.

There is a phosphor coating on the inside of the face of the tube, and the electron beam, when it strikes this coating, excites the phosphor, causing it to glow. If the phosphor coating is very fine and the beam is closely focused, a very tiny and quite brilliant dot can be created. If the beam is moved vertically, by deflection, then a series of dots, resulting in a vertical line, will be formed. How long the line will be visible on the face of the tube will depend upon the "latency"

of the image—the "decay" time of the phosphor. For some applications, where the same image has to remain for a period of time, the image is "refreshed" at, perhaps, 60 cycles per second, by recycling the same bombardment of electrons. But for typesetting purposes, a low degree of latency is desired, and the initial image must be sufficiently brilliant to photograph well.

Such tubes generally fall into two classifications: one is like a TV tube which is so designed that the beam will sweep the tube in a predetermined pattern, thus creating a network of horizontal lines in a "raster scan" pattern, very close together. The other type of CRT will direct its beam in a random manner, in order to "draw" on the face of the tube, depending upon the emission and deflection signals it receives. It is this latter type of tube which is to be found in the third-generation phototypesetter.

Distortion.—Since the beam has to travel further to reach the edges of the tube, the beam is somewhat more diffused (out of focus) when it arrives. Also, because of the angle at which it strikes, there will be an additional spreading of the dot into an oval shape. Hence characters produced anywhere other than in the center of the tube are subject to some degree of distortion unless there is a correction factor introduced into the electronic circuitry, causing more or less deflection and changing of focus patterns. But applications involving high-quality reproduction nevertheless seek to avoid the use of the outer perimeters of the tube, especially the use of the far corners. The distortion around the corners is called "pincushion" distortion.

Compensating circuitry is also built into the electronics to correct for distortion in all models of CRT typesetters. The larger the area of the tube used, the more complex the circuitry. The larger the CRT, the larger the "good" center area of the tube and the less the distortion problem. But the larger the tube, the more expensive it must be and the more difficult to build. In the 1960s, Columbia Broadcasting Laboratories was involved with Mergenthaler in the design and construction of the Linotron 1010 largely because the full-page feature of that device required an unusually large CRT of exceptional quality, and it was this technology that the Laboratory could contribute.

The Linotron 505

While the first third-generation photocomposition device was either the Linotron 1010 or the VideoComp 820 (depending upon whether the historian goes by date conceived or date delivered), it is more logical to begin our discussion with a description of the Linotron 505, since its operating principles are closer to second-generation machines.

The original "505" was developed by Peter Purdy and Ronald McIntosh—two Englishmen whose work was financed by K.S. Paul. Their development effort began in the year 1963. The initial concept of the machine was not unlike that of the (later) Photon Zip in that 256 flash tubes were to be employed for the purpose of character selection. But as the months went by the product definition

changed. They decided instead to *scan* photographic images, at first planning to use 64 indexing tubes, each of which would be able to scan four characters. Finally the machine's basic structure emerged and a prototype was revealed at the DRUPA show in June, 1967, many pounds and several years off target.

That prototype, capable of setting only in sizes up to 14 points, was installed at the Portsmouth (England) *Evening News* in February, 1968. But Mergenthaler, the American typesetting manufacturer and vendor, reached an agreement to acquire the company and its machine at, or immediately following, the DRUPA show. However, the Mergenthaler/K.S. Paul agreement was conditional upon the British company's ability to extend the maximum point size to 28. This was in fact accomplished, although it made necessary the revamping of the entire optical system. The new machine was ready, then, to be unveiled by Mergenthaler as the Linotron 505 (one half as powerful as the 1010?) in Chicago in the summer of 1968, at PRINT '68. By this time, Mergenthaler had acquired a 74% interest in K.S. Paul & Associates—an interest which was later increased to 100%—and Purdy and McIntosh decided to leave the company in order to form a new venture of their own. Gradually, Mergenthaler began to exercise more and more control over the product as they interested themselves in quality, reliability, and font development.

The 505 represented a distinctive approach to CRT phototypesetter design. Perhaps the two most important facts to remember about the machine are that it uses photographic grids as a character storage medium, and that it uses a traveling lens system to lay down characters—or, more precisely, the vertical strokes that go to make up characters.

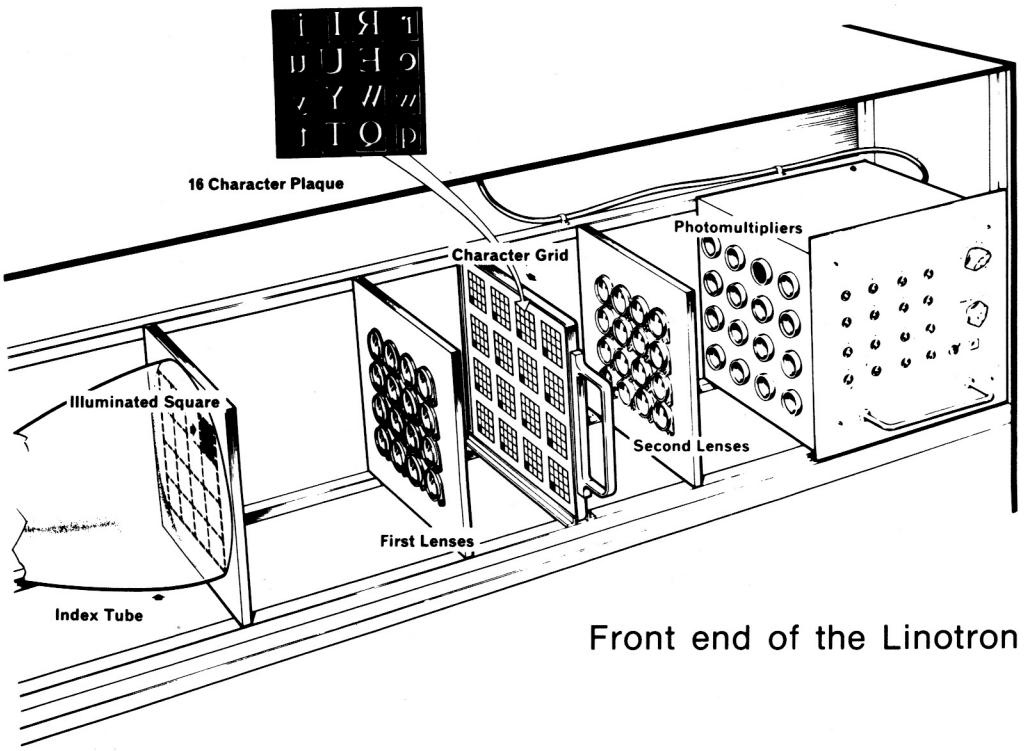
The use of the traveling lens eliminated the problem of distortion correction on the face of the CRT. Only one vertical stroke or slice of the character is generated and reflected onto a traveling mirror or prism, which deflects that stroke onto the photographic film or paper. Unlike second-generation machines, where the entire character would be positioned by a traveling mirror or prism, here the escapement takes place one stroke at a time. As each stroke is generated, a light-sensing raster reads the position of that stroke and calls for the scanning beam to transmit another. A succeeding vertical slice is then generated. The mirror travels across the line at a constant speed, calling for consecutive vertical strokes—650 times for each inch of travel, in the coarser resolution, or 1300 times for each inch in the finer resolution.

Characters are stored as photographic masters on four grids, each of which contains 238 images. In addition, a separate "pi plaque" of 64 characters is provided.

Each of the four grids consists of an array of 16 sections. Directly in front of this grid are sixteen lenses, and in front of them is a character selection scanning tube or CRT, which is capable of "indexing" a particular location on the tube—that is, generating raster strokes which are picked up by all the lenses in such fashion that a particular character location is scanned on *each* of the 16 grid sections. However, behind the grid there are sixteen photomultipliers, only one of

which would be "turned on" so that only one character would be scanned and stroked. This stroking information would then be conveyed to the output tube which, in turn, would generate comparable strokes, each of which would be laid down by the traveling mirror. Between the mirror and the film plane there is a lens which reduces the stroke image at a ratio of 9.5:1, thus enhancing the quality of the output. *Later models of the 505 substituted a tiny traveling CRT output tube for the traveling mirror*, but the principle of selecting and scanning and transmitting character information on a stroke-by-stroke basis to a traveling stroke-laydown device remained.

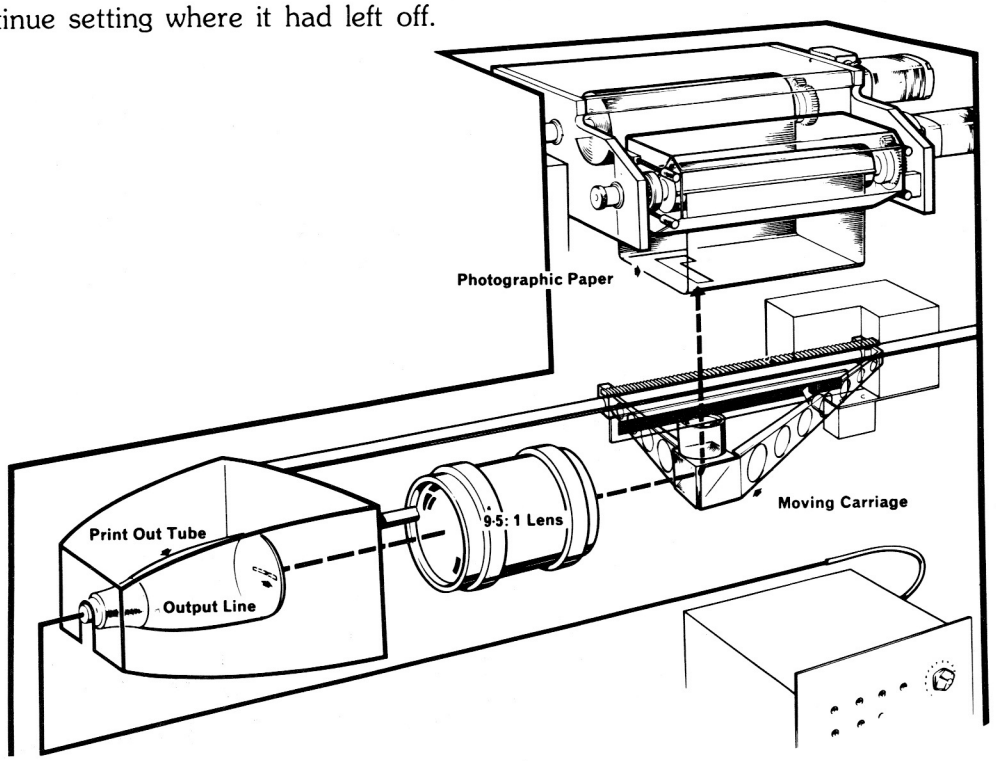
Despite the fact that there are lenses in this system, the *sizing* is nevertheless electronic in character. In other words, the length of the stroke on the output tube and the number of strokes per character determine the size of the output character. At the resolution of 1300 lines per inch, for example, a character 36 points wide would consist of 650 vertical strokes, whereas a character 18 points wide would consist of 325 vertical strokes. Electronic circuitry would "know" the desired size of the character and consequently how many slices of that character would have to be scanned.



Front end of the Linotron 505 photo unit

Character images are stored on a grid which contains 16 plaques, each, in turn, consisting of photographic masters for sixteen characters (although only 238 characters are represented in all). In front of this grid, to the left in the diagram shown above, is mounted an index or scanning cathode ray tube. Sixteen lenses focus the face of this tube onto each of the 16 grid plaques. Sixteen additional lenses behind the grid focus each of the 16 grid plaques onto one of the 16 separate photomultipliers. The face of the Index CRT is divided into imaginary squares. To scan a character on the grid, the 505 will generate a series of scan lines in one of these square areas on the tube. These raster lines are projected through "first lenses"—all 16 of them—in order to stroke one character from *each* of the 16 photo masters on the grid. These strokes all pass through a second group of lenses, but only one character is then selected for transmission to the print-out tube, because only one photomultiplier is activated.

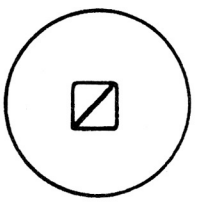
What the initial 505 achieved. By way of *character store,* 238 typographic characters are available at one time in a grid. Four such grids can be mounted. New grids can replace old ones by simple manual insertion. All characters on one grid are instantly available. The grid windmill, which can shift any one of these additional grids into position, requires 2.8 seconds to move from position 1 all the way around to position 1 again. A movement from position one to position 3 takes 1.4 seconds. *The moving carriage returns to the start of the line whenever a grid change takes place and remains there until the change is complete.* Then it will "move out" over the material already exposed in the line and automatically continue setting where it had left off.



The heart of the "print-out" end of the photo unit is a carriage which travels along a rail underneath the photo media to be exposed. On the rail is a fine-line grid with a calibration line every 1/18th point (making approximately 1300 per inch). Every calibration line, or every alternate calibration line, depending upon whether the 505 is running in "high" or standard resolution, causes the carriage to send back a signal which calls for the next stroke. A scan stroke is then generated on the index CRT and the CRT beam on the output tube shown in the diagram above is deflected in "sync" with the scanning stroke. When the photomultiplier behind the character to be set senses the light from the scanning beam (*see preceding diagram*) it signals the beam on the output to turn on. When it no longer senses light from the scanning beam it signals the beam on the output tube to turn itself off. The character image is thus transmitted to the photographic film or paper in a stroke-by-stroke fashion as the carriage moves across the measure.

"Pi plaques." In addition to the characters instantly available on one grid, a total of four plaques of sixteen characters each—making 64 in all—can also be positioned in such fashion that their images are always accessible with no time delay, by virtue of an arrangement which deflects the raster scan, and redirects it to one of the photomultipliers. These 64 graphic symbols are usually "universal" characters which are font-design independent.

Electronic manipulation—slanting. Although the first thirty or forty 505's performed the slanting or "obliquing" function by virtue of the juxtaposition of a distorting lens, this "slow function" which requires the time delay necessary to shift the lens into place and remove it again, was eliminated in favor of electronic *obliquing*. It became possible to slant any character to the right at a 12° angle (8° in Germany, by their preference) through the use of electronic means. In other words, the single stroke generated by a single scan of the index tube appears in the CRT "window" in the manner illustrated in the margin. This, of course, produces an obliqued roman rather than a true italic. Italics posed some problem as there is no way of backing up the 505 carriage on the fly to get characters to kern. By kerning in this context we do not mean a change in the side bearings of specific characters to achieve a better fit, but merely a design which permits one character to flow under or over the preceding or succeeding characters, whatever they may be. The problem of generating "true" italic was solved by "straightening up" the italic masters by means of an anamorphic lens when they were photographed to make a grid. When typeset, italic fonts are then electronically slanted back to their original slope. In this way the lower case "f", for example, will kern under the character which precedes it and over the character which follows it.



afi

In practice, in order to conserve valuable grid positions, usually only the lowercase characters are straightened up and then slanted back into shape. It was decided that "true" italic capital letters were not necessary. Hence, italic numerals, punctuation and capital letters are merely roman forms which are obliqued.

Electronic manipulation—changing point sizes. The Linotron 505, as well as the subsequent C and TC models, provided the user with a choice of fifteen sizes of type. You could specify any fifteen with the range of four to 28 points, in one-half point increments. After some fifty machines were delivered, a "wide-range" version was introduced as an option. This option makes it possible to increase the length of the stroke on the output tube so that it can become four times as long, thus adding substantially to the power of the 505. You can have your basic 15 sizes plus another set four times larger (although there is, inevitably, some duplication of sizes). To achieve the wide-range capability it was necessary to move the print-out end of the photo unit into a different box where it could take up more room. Thus the "regular" 505 consists of two boxes (one for the "logic" unit and one for character generation) while the wide-range model has three—the logic unit, a second for scanning, and a third for character laydown. In the wide-range unit the collimating lens shown in the diagram on the preceding page is replaced by a mirror which sits below the lens carriage and, from its approximate angle of 45°, deflects the beam stroke through the traveling lens.

The combinations of sizes that a user might select are set forth at the top of the next page. The reader will note that, despite the ability of the "basic" machine to set 28 point type, the user is better off, if he wants the "wide range" or 4X feature, to select only smaller sizes and to rely upon the wide-range capability to provide all sizes over, say, 18 points. In this fashion he will duplicate fewer sizes and obtain the wide-range capability to provide all sizes over, say, 18 points.

Linotype-Paul and Mergenthaler did not intend that the machine, with its wide-range feature, should go above 72 points, but their customers have set 96 point type which they apparently considered satisfactory. Resolution in the 4X range is 650. It is not possible to go to 1300 because the slower speed of carriage movement is required in order to get sufficient time to create the longer strokes demanded of the larger sizes.

Basic 15	*Wide Range*
4	16
5	20 (duplicate)
6	24 (duplicate)
7	28 (duplicate)
8	32
9	36
10	40
11	44
12	48
14	56
18	72
20	80 (not contemplated)
24	96 (not contemplated)
28	Not possible

Note that the 4X sizing is achieved by electronics and not by magnification.

Electronic manipulation—changing base lines. Another electronic character manipulation feature, initially hard-wired into the machine, is the ability to raise or lower the base line (still within the CRT window) in six steps without the need to move the film forward or to reverse it. Such deflection makes it possible to create, from any typographic image, a superior, a superior to a superior, an inferior, or an inferior to an inferior. When the 505C came along this feature was taken over by software, and it became possible (as an option) to raise or lower base lines by one-half point increments anywhere within the "window." This window, by the way, has the following limitation: the maximum stroke can be 120 points high. If you were setting a 96-point character you would be able to move it below the base line by about eight points and above the base line by about sixteen points. On the other hand, a four-point character could be raised or lowered through the whole range of 120 points—a facility of some importance for math or other multi-level setting.

Electronic manipulation—changing set sizes. As with most other CRT typesetters, the 505 is able to treat character height and width as two independent variables. For example, photographic masters may be scanned as if they were twelve points high and six points wide, or vice versa. This distortion capability is limited, however, to the basic fifteen size elections. Thus if you purchased a machine which has the ability to set 8, 9, 10 and 12 points, you could set an 8-point character 8, 9, 10 or 12 points wide, or a 12-point character, 8, 9, 10 or 12 points wide, *etc.* This capability proved to be extremely useful for newspaper applications, especially for display headlines.

Resolution

The 505 was built to offer a resolution of either 650 lines to the inch or 1300.[1] As previously stated, the resolution is determined by the frequency with which the moving carriage calls for a new scan line.

1. More precisely, the calculations are as follows: If one point equals 1/72 of an inch, with an 18-unit system (and therefore 18 units to the point) there can be 1,296 scan lines to the inch.

CRT typesetters which generate their images by scanning photographic masters invariably lay down character strokes at a constant resolution. In this case, as we have mentioned, the resolution is 650 or 1300 lines per inch. On the other hand, most CRT typesetters (but not all) which generate their character images from digitally-scanned masters, vary the resolution or number of strokes per inch in order to achieve different character widths to correspond with changes in point sizes. On the whole, it would appear to be advantageous to try to maintain the same "resolution" regardless of point size, but in actual practice a variation in "resolution" is not as important as it would at first seem since there are methods of compensating for the variations in character density which might otherwise result.

Speed

The speed of the 505 is first of all a function of character resolution. At 650 lines per inch (when not using the 4X feature) you can set approximately 160 twelve-pica lines per minute regardless of point size, assuming no grid changes or other "slow" functions. The forward speed of the traversing lens is 14 inches per second at 650 lpi and seven inches per second at 1300 lpi. The carriage return (except on the TC model) takes place at 28 inches per second. The TC model, in setting type with no more than ten-point leading, can set almost twice as fast, *since it sets backward as well as forward,* or at the rate of more than 290 lines per minute.

If, however, you had two grid changes per line, from grid #1 to grid #3 and back to #1 again, speed would be reduced by 2.8 seconds per line, plus the time it would take to re-scan the portion of the line already set, which would mean that under such (unlikely) circumstances, throughput would be reduced to about 20 lines per minute. For speed in setting, therefore, it is distinctly advantageous to set only from the characters in one grid plus the pi plaque, and to vary style electronically (by slanting, distorting and sizing) rather than mechanically.

Quality of the image

Images created by the scanning of photographic masters will generally not be as "clean" as images of comparable resolution generated from digitized masters. This is because when you scan a straight line precisely along its edge you will get random effects. On one occasion the scanning beam will sense an image and on another it will sense no image. This tends to create "burrs" and "notches" in the character outline, but such deviations are usually not nearly so substantial as deviations attributable to different molding effects from the same

hot metal matrix, and are certainly nothing like the variations from casting "identical" characters from the circulating matrices of the linecaster.

There are two other peculiarities of the method by which the 505 scans character images which should at least be mentioned. The first of these relates to the index or scanning tube: the correction for pincushion distortion on this tube is not complete. Therefore the scanning stroke is not totally straight, especially near the outside corners of the tube. This can introduce a slightly curved or bent appearance to some characters—an effect which Mergenthaler minimized by locating characters which would suffer from this problem near the center of the grid plaques, and those where it could hardly matter—such as periods, commas and the like—on the far corners.

The second peculiarity relates to the photomultipliers themselves: it proved to be impossible to get two photomultipliers which behaved exactly in the same fashion or which aged precisely in the same manner. If one of the 16 photomultipliers were more sensitive than the others, 1/16th of the characters on the grid would reproduce with more "weight" than the others on the phototypeset output.

Incidentally, the photographic masters are 14 points in size, but they could be selected from eight, twelve or eighteen point designs.

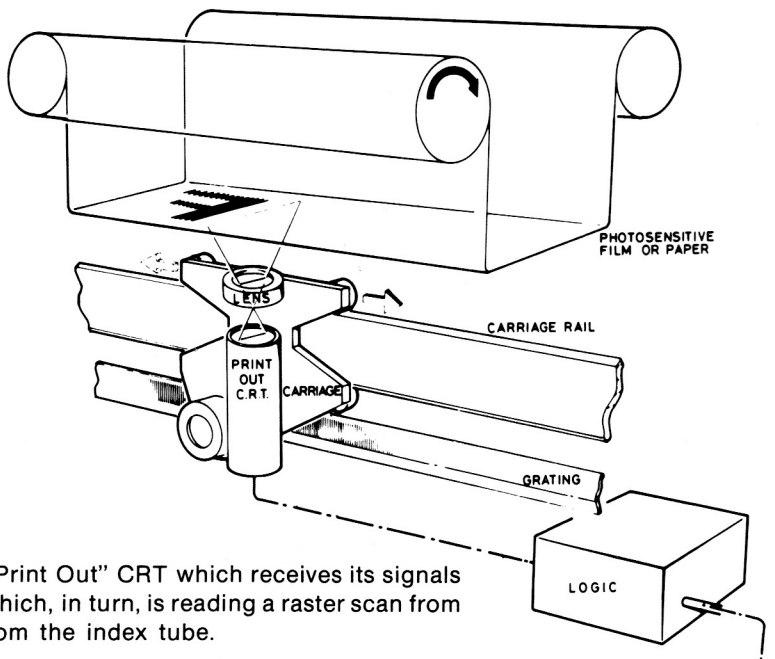
The "C" and "TC" models

After manufacturing almost 100 of the 505's, Mergenthaler and Linotype-Paul replaced the "hard-wired front-end" with a programmable minicomputer—a Honeywell 316. This was to make it possible to set lines of type "on the fly"—that is, from right to left as well as from left to right, and thus reduce the idle time required to move the prism or mirror back to the beginning of the line. By using a minicomputer it also became possible to enhance the "intelligence" of the machine by adding still more "core" or "memory." (These concepts will become clearer after our discussion of computers in the next chapter.) Thus, with sufficient memory the 505C could be operated as a "stand-alone" typesetter, which could make end-of-line decisions, including hyphenation. The designation "C" means that the typesetter now contained a computer, and "TC" meant that sufficient memory existed to permit the machine to set alternate lines backward.

Still later, Mergenthaler/Linotype Paul addressed the problem of how to set a longer line. The TC has the ability to set to a 64-pica measure, but for full newspaper page composition (assuming that multi-column material can be typeset) a width of 100 picas is necessary. In order to achieve this the design of the machine was changed and a small, traveling CRT tube took the place of a fixed output tube and a traveling lens or prism. Consequently, on the 505TC there is a tiny tube, capable of generating a vertical (or slanted) stroke which, in turn, is somewhat magnified, as is shown in the illustration on the facing page. The 100 pica model, with the traveling CRT, also contained other engineering improvements which made possible a further increase in setting speed, and enhanced reliability.

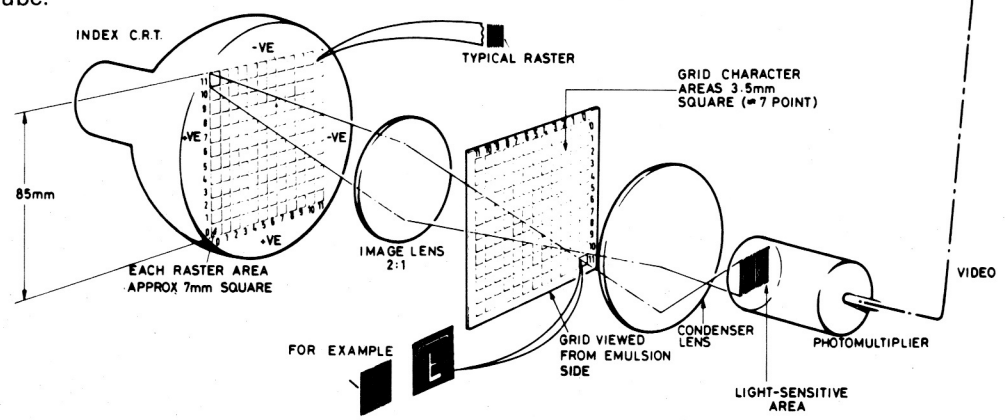
The Linotron 303

As we have already suggested, the "windmill" arrangement for providing additional font masters, although it substantially enhanced the character repertoire of the machine, slowed it down considerably. Consequently, in 1974 the company announced the "303" which replaced the 238-character grids with smaller grids of 144 characters each. Two models were offered—one offered eleven such grids with a rotating container, and the other was designed to permit the mounting of 24 grids of a similar size. The machine would, on signal, access any one of the grids by rotating the large disc in which they were mounted. Grid changes take from .2 to .5 seconds, and, of course, as with the four-grid 505, individual font grids are interchangeable.



The Linotron 303 uses a "Print Out" CRT which receives its signals from the photomultiplier which, in turn, is reading a raster scan from the strokes transmitted from the index tube.

Note the grating on the carriage along which the Print Out CRT travels. The grating positions are sensed by a photomultiplier (not shown) and it is this signal which calls for another stroke from the index tube.



By the time the Linotron 303 reached the market, in late 1974 or early 1975, the interest had clearly shifted away from CRT devices with photographic masters to machines which kept all of their fonts on-line in a digital form. Mergenthaler Linotype-Paul's research facility was itself developing such a typesetter in the Linotron 606. While the 505 and the 303 were still widely used, it seemed clear that the digital CRT typesetters were attracting the most attention. However, there was one significant exception to this generalization, and that is the Compugraphic Videosetter which, because of its high speed and extremely low price, carved out a rather unique market for itself. It is appropriate, therefore, to consider the Videosetter before we turn our attention to those CRT typesetters which utilize digital character storage techniques.

The Compugraphic Videosetter

The Videosetter was first demonstrated in June 1973. Thus ten years had elapsed from the first Purdy-McIntosh effort that gave rise to the Linotron 505 and the subsequent 303. And compared with the 505, the Videosetter represented some startling innovations. These resulted in a product price which broke away from the established CRT pattern as dramatically as Compugraphic's introduction of the 2961 and 4961 models of second-generation typesetters in 1968 broke away from the then established prices of other similar devices. At a time when no other CRT typesetter was available at a price under $100,000, Compugraphic announced the Videosetter I at $32,950, and subsequently reduced the price of an improved product to $21,950!

Initially, the Videosetter was a very limited machine in terms of its character repertoire, maximum point size, and setting measure. Only 106 characters were available at a time, only certain sizes up to 36 points could be generated, and a line no wider than 27 picas could be set. This was clearly intended to be a machine for the newspaper industry, and especially for the composition of single-column measures, along with two-column display ads. Subsequently, however, some of these limitations were overcome.

Principles of character generation. The accompanying diagram should make the principles of the machine quite clear. A grid containing 106 photographic masters is mounted below a light-diffusing lens so that all 106 characters are simultaneously illuminated, and the entire grid image is reduced by a ratio of 2:1 and brought into focus on the face of an *image dissector* tube. It is this tube which provides the key to the system.

> "One of the most extraordinary devices is the image dissector tube that was invented by Farnsworth during the early days of television," wrote Ed Eberhardt, chief scientist with ITT.[1]
>
> "In recent years it has interested more people and is being used to solve . . . tough technological problems. For instance, using the dissector the computer can be provided with quantitative optical information . . . at any area it wants to look at. In other words, it has random access capability. Perhaps the biggest change which we made 6 or 7 years ago was the addition of an acceleration

1. *Electro-Optical Systems Design*, January 1973, p. 27.

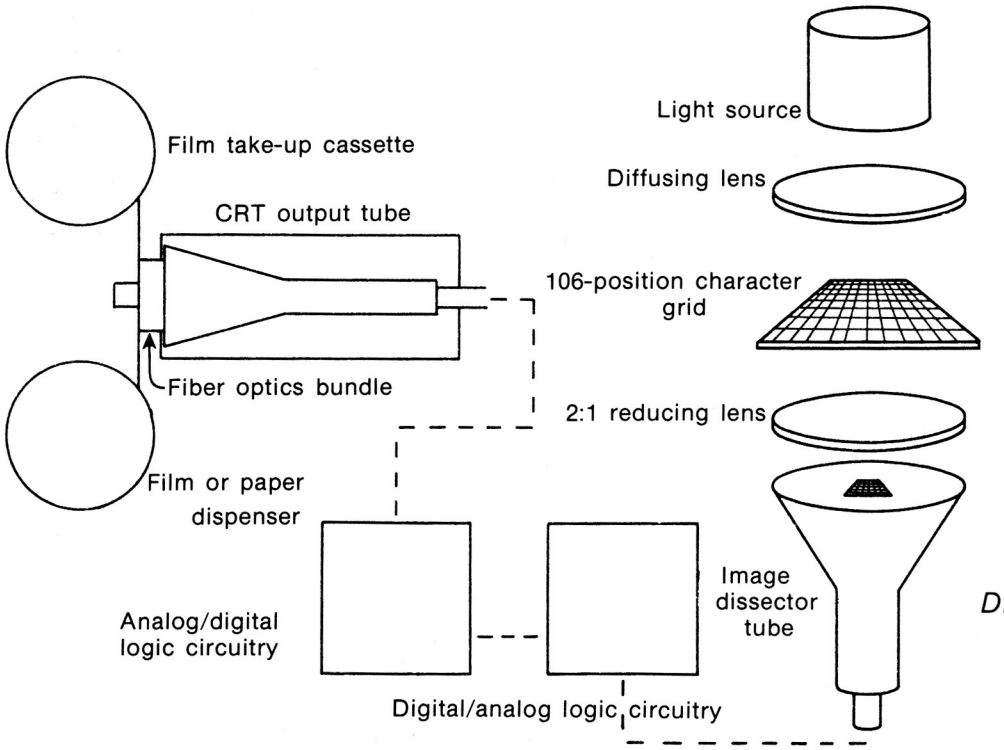


Diagram of optical principles of the Videosetter

mesh to the dissector. By putting in such a mesh, the resultant electrons are injected into a field-free region where they could be focused and deflected with a minimum distortion and aberration. This resulted in a quantum leap forward in terms of what a detector could do."

The scanning beam of the image dissector tube is able to select from the grid pattern one character, and to scan that particular character and to transmit its features despite its minute size. The image on the grid itself represents a 12-point master size. It is reduced through the focusing lens to six points. Nevertheless, the character is scanned or swept by the scanning beam at a frequency which provides a "resolution" on the film plane of 1300 lines per inch.

The signals which are "read" during the scanning process are used to turn a writing beam on and off on the output tube. But the length of the stroke on the output tube is a function of the size of the character desired to be typeset, just as the number of times the character is scanned on the image dissector is also a function of the desired character width. In other words, if the output character is intended to be 36 points, and the character being scanned is an em wide, then the character as projected onto the image dissector would be scanned 650 times. If it were to be an 18 point character it would be scanned 325 times, and 325 vertical strokes would be generated on the output CRT character generator.

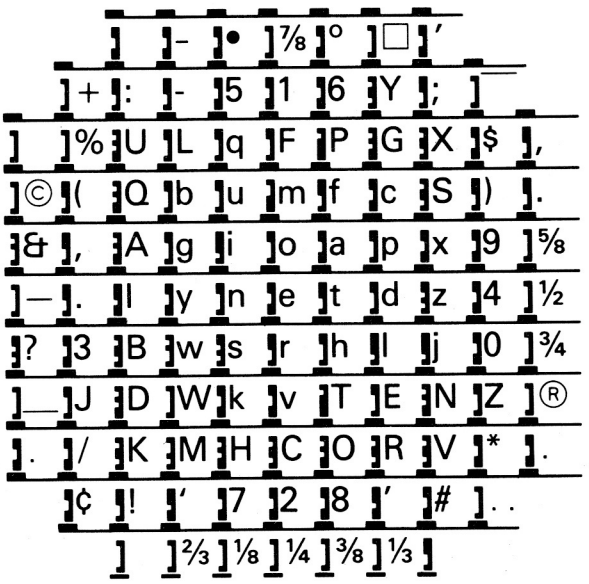
Just as the logic of the image dissector circuitry must "know" what character to scan, so the logic of the CRT character generator must know where, on its

tube, to lay that character down. This process is perhaps not as complicated as might be imagined since all characters are generated in consecutive order.

The original Videsetter's output or character generating tube was only 5" in diameter, with an "aperture" or "window" within which characters could be stroked which was only one-half inch high. Hence the maximum line length was 27 picas (4½ inches) and the maximum character height was 36 points.

A larger image could have been generated on the output film or paper if Compugraphic had elected to interpose a lens between the CRT output tube and the film plane. This is quite commonly done, but Compugraphic elected, instead, to use a CRT tube with a *fiber optic face plate*. In this fashion it saved the cost of a high-quality lens. A fiber optic face plate consists of a bundle of fiber optic tubes which are bonded to the face of the CRT character generator. The photographic paper comes into physical contact with the outside ends of the fiber optics bundle while the inside ends, which are coated with phosphor, are mounted directly into the tube itself, so that they can be excited by the electron gun.

The Videosetter grid. If we examine the grid layout, certain other features of the machine immediately become obvious. Note how the most common characters are positioned in the center of the grid, where the scanning capabilities of the image dissector will provide the best results and the minimum amount of beam deflection will be required in order to access these images. Note, also, that there are no characters to be found in those areas where distortion is most troublesome, and those closest to the corners are characters where distortion would matter the least. Nevertheless, it is important for the tubes—both the image dissector and the character generator—to contain appropriate circuitry to correct for distortion, and this is done not by dividing the tubes into segments for correction purposes, but by devising one hardwired map (with adjustable controls) to regulate beam deflection and dot intensity to compensate for the inherent limitations of any such CRT device.



When the grid is mounted in the machine, the machine is "primed." This means that you touch a button which causes the image dissector to "read in" the diagrammed width values for each character, and to store them in its memory. Thereafter these width values are no longer accessed or scanned by the image dissector, for now the machine "knows" what each character's width value is. You will note that Compugraphic uses an 18-unit system. There are white squares out of a black bar wherever there is a value to be read. Here the bar on the left has a maximum of white space. Reading from the top you get 8+4+2+1, which equals 15. Compugraphic adds three units to all such values, making our first total 18. The second bar shows 8+1 (plus 3, of course), making a total of 12. The third shows 4+1 which, with 3, equals 8 units.



Electronic manipulation. So far we have seen that the Videosetter can scan and reproduce 106 characters. We have intimated that these characters can be *sized*. The Videosetter contained circuitry to read characters from the image dissector tube as if they were 5, 6, 7, 8, 9, 10, 12, 14, 18, 24, 30 or 36 points, and to generate vertical strokes of sufficient length—and a sufficient number of strokes of constant resolution—to output characters of the desired size. But other electronic games are also possible. The original machine offered the ability to stroke a character vertically at one point size on the output tube, while reading its width horizontally at another size, thus achieving the ability to expand or condense the appearance of the type face, so far as width or set size is concerned. Initially, this was only possible for one expanded and one condensed version of each basic point size, but subsequently the electronic circuitry was modified so that type could be sized in one-point increments. Thus, in theory, a character 36 points high could be generated as if it were any set size from 5 to 36 points.

In addition, characters can be obliqued, to simulate italics, or emboldened, to simulate a bold face. Consequently, one 106-character grid could create many different character images—not only with respect to sizing, expanding and condensing, but also with respect to emboldening and obliquing.

Subsequently, Compugraphic enlarged its Videosetter line, offering several models, of which the Videosetter Universal is the most flexible. This machine provides for the mounting of eight grids (four in each of two grid turrets), and it outputs onto an eight-inch tube instead of a five-inch tube, thus making it possible to set a 45-pica line length, and to generate characters up to 72 points in height. With so many different grids "on line," users are now less inclined to take recourse to the "emboldening" and even the "slanting" or "obliquing" options since, if they prefer, they can mount character images which are already designed as true italic or true bold faces.

The spot size on the face of the output tube, and hence on the film or paper, is approximately one mil (one one-thousandth of an inch). The 1300 line-per-inch resolution allows for sufficient (but not excessive) dot overlap. The speed of the machine is approximately 45 column inches per minute. This is equivalent to 360 11-point lines of 9-point type per minute. Paper is advanced after each line has been composed. Compugraphic has also built "look-head" logic into the machine

so that as it reads the incoming text, if it discovers that the next grid required is on the alternate turret, it can rotate that turret into position before it is required, in order to minimize time loss from the mechanical motion of the turret.

The Linotron 1010

In order to complete our run-down of CRT typesetters which generate their images from photographic masters, we should mention two other machines. The first of these is the Linotron 1010, which was the first CRT machine ever to be commissioned. Five of these machines were built by Mergenthaler (with the assistance of CBS Laboratories), and the first of these was accepted for delivery in September, 1967. Two of the 1010's were installed at the U.S. Government Printing Office, two were sold to the Wright-Patterson Army Air Force Base, and a fifth, while used briefly for commercial purposes, was ultimately cannibalized for components.

Like the 505, characters are generated on the face of the CRT from scanned photographic masters. A four-inch square glass grid, containing 255 characters and a test pattern, is positioned so that its store of characters can be scanned. A windmill-like arrangement is available to permit shifting from one grid to any of the three alternate grids. Characters are not scanned by a flying spot, but the result is the same. A very sophisticated, ingenious and expensive solution was developed, whereby a "cloudlet" of electrons is swept over a stationary aperture causing an analog signal to turn the current on and off. This signal is then used to generate the character image on the face of the CRT.

The 1010 output tube is a "full face" character generator. The machine sets a page at a time and then moves the film or paper in order to compose the next page. This means that the face of the tube must be large enough to contain the entire page (8¼ x 10½ as enlarged 2:1). This makes for a very expensive tube, with complex correction circuitry, and it also implies an exceptionally high replacement cost.

Due to the full-face characteristics, the positioning of images on the face of the CRT is controlled not only in a horizontal sense, but in a vertical sense as well. It would theoretically be possible (if the computer software were sophisticated enough) to set characters from *one* grid, wherever on the page they might be required, and then to shift grids and "spray" additional characters wherever they were needed—special symbols, display heads and the like.

Character density (resolution) is the same regardless of point size. In this respect the 1010 is like the 505 and the Videosetter. But unlike these machines, character lengths and widths cannot be controlled independently. There is some limited capacity to condense, but it is not sufficient to specify set widths as variables independent of their point sizes.

The speed of the 1010 is given as 1280 characters per second for five point type, 800 for eight points, 620 for ten, 400 for fourteen, and 270 for eighteen

points, representing a constant rate at which the strokes of characters are generated. In addition, the machine will set twice as fast in a "proofing" mode because it generates only one out of three vertical strokes.

Beside the 1010, only the VideoComp presently offers the full face capability (as an option), in the U.S. market, as does the Digiset in Europe. Other devices compose type only in the "aperture"—usually one line at a time.

The Crosfield Magnaset. Passing brief mention should be made of one other CRT typesetter which created its type by means of scanning photographic images. This was the Crosfield Magnaset, developed by Crosfield Electronics, in London. What was unique about this device was that photographic masters were positioned in rows around a rotating drum, and when a character was desired by the output module that particular character would be scanned and electronically re-created on the face of the CRT character generator. The purpose behind this design was to eliminate the delay in setting occasioned by the necessary grid movements of the Linotron 505. However, by the time it was nearly perfected (1973-74), the digital font storage method had grown in popularity and acceptance and this project was therefore abandoned.

The digital machines

We shall now discuss CRT typsetters which do not derive their character patterns from the scanning of photographic masters *on line*. That is to say, the desired characters are not stored photographically in the typesetters; rather, a digital representation of each character to be typeset is stored in memory, where it temporarily resides. The original art work may well have been created by scanning, in a manner somewhat similar to that used by the photographic-image CRT typesetters. But the scanning process would have occurred only once, in order to generate the digital pattern which would then, itself, be stored, duplicated and reused by many different typesetters, wherever located, whenever that character image was desired and the digitized font master had been purchased by the user.

It is not the output characteristics of the digital machines which tend to make them unique: it is the use of digital character storage. Characters images are stored "as stroke information." A font of characters, or a subset of a font, or even an individual character, is brought up into the "memory" of the CRT typesetter on call. Such fonts are now customarily stored on a moving-head disc which is a peripheral to the typesetter, although initially fonts were loaded into the memory of the typesetter from magnetic tape, one font at a time, as part of the text stream. Generally speaking, a character consists of a series of vertical strokes (just as if they had been scanned), and these strokes are painted on the output tube from the bottom of the character to the top. Hence, if one wanted to describe a character, one has only to record a series of "start/stop" points which indicate where the electron beam character-generator is to be turned on and off. The beam circuitry would continue to sweep the aperture of the tube in regular strokes (but not in a "raster" pattern since the beam can be deflected both vertically and horizontally by signal). The deflection circuitry generally moves evenly across

the tube's aperture, however, especially when painting a particular character. When that character has been completed deflection electronics will position this beam where the next character to be set is to be located. But the electronic bombardment only occurs when the gun is turned on to spray the electrons which will excite the phosphor coating on the inside of the tube. Thus, what is significant here is the signal to turn the beam on and off during the course of the deflected stroking movement. Since the timing of the beam movement is known, a signal to start the beam stroke, to hold it on for the equivalent of four (hypothetical) vertical cells, to turn it off for three more, to turn it back on again for two, and then to turn it off and return to the starting point, one stroke to the right in order to begin another sweep, would permit the generation of a light pattern which would comprise a vertical "slice" of the desired character.

If we consider the task of generating a ten-point character true-size, and if we wish to create this character with a resolution of 720 strokes to the inch, then since there are 72 points to the inch, a ten-point solid em would consist of 100 strokes. A lower-case letter "a"—if it were one-half of an em in width—would therefore consist of 50 vertical strokes. It appears to be most convenient to express character widths in terms of a 100-unit system. Thus our lower-case "a" would have a width of 50 relative units and an absolute value of five points at true master size.

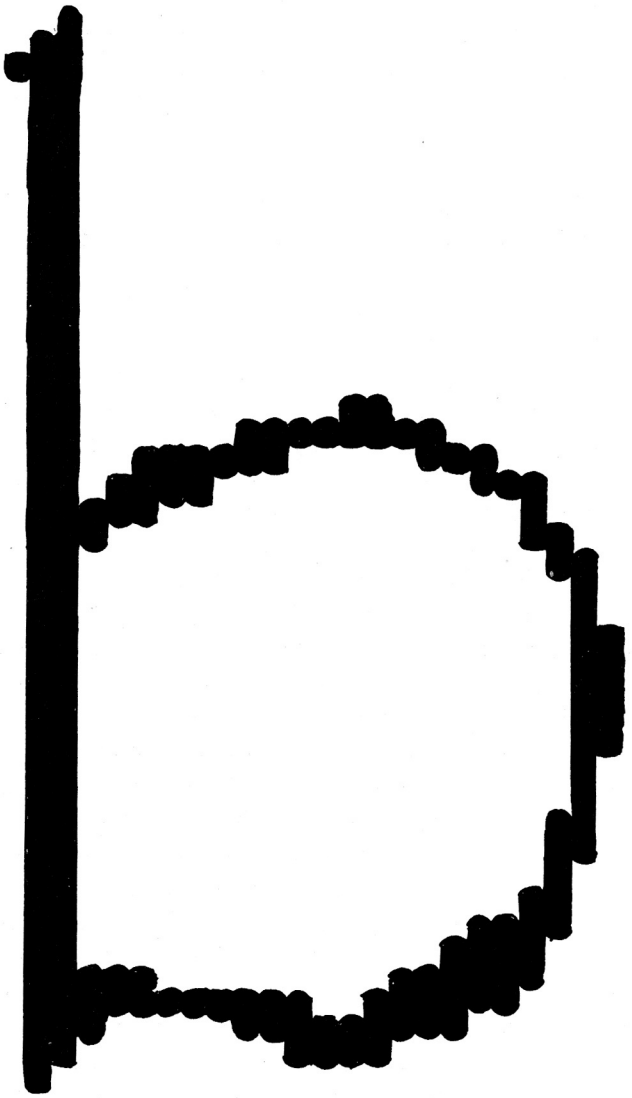


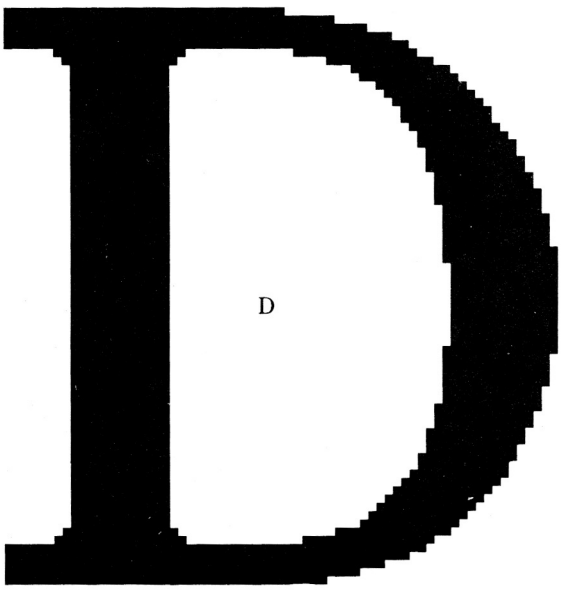
Thus, if the average character were, in fact, an "en" in width, and the master were ten points, then 50 vertical slices would be sufficient to describe that character. But the number of times the beam would have to be turned on and off during the stroking cycle could be quite considerable. Perhaps the "worst case" would be the "at" sign, for, as you can see in the margin, there would be a number of locations where the starting and stopping points for beam exposure would have to be specified. A more typical case might be represented by the letter "b" shown on the next page. To simplify the illustration this character has been drawn, rather roughly, on a scale of 50 strokes to the em rather than 100, and the locations for turning the beam on and off are also shown, with the horizontal identification of "on/off" locations indicated by numbers 0 through 49.

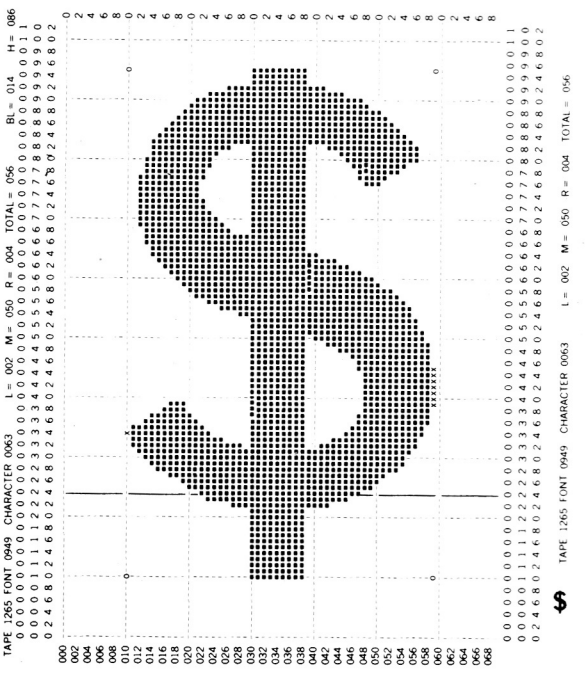
An inspection of this crude character makes one wonder how it is possible to create satisfactory graphic images from a grid pattern of dots or strokes. A slight dot overlap is helpful to give the character a better shape, and the finer the stroking pattern the better the representation will be. To illustrate this point we have also shown the letter "D" as it would appear in type and as it looks as digitized "artwork." Diagonals are obviously more difficult to represent than vertical lines, and graceful curves may be hardest of all, particularly if they are subtle. It is for this reason that CRT type is most successful when it has been "edited." The mere digitization of art work usually (but not invariably) leads to inconsistent and less-than-satisfactory results, just as the scanning of photographic images usually produces character shapes which are not as clean and well-formed as those which have been digitized first and then edited. One reason this is so is because the scanning of the edge of a vertical line often results in a ragged appearance as the scanning beam becomes confused and turns itself on or off. Another reason is

Right: The capital letter "D" is shown in its digitized form (from Autologic), with the master size also shown.

Below: The lower-case "b" with its starting and stopping points defined below it.







Above: A graphical representation of a dollar sign, as digitized by Autologic. The start and stop points and the positions of each stroke may be ascertained by the coding. The latter does not correspond, however, with Autologic's stroke encoding storage technique, but is purely for analysis in editing.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	
49																											49
48																											48
47																											47
46																											46
45																											45
44																											44
43																											43
42																											42
41																											41
40																											40
39																											39
38																											38
37																											37
36																											36
35																											35
34																											34
33																											33
32																											32
31																											31
30																											30
29																											29
28																											28
27																											27
26																											26
25																											25
24																											24
23																											23
22																											22
21																											21
20																											20
19																											19
18																											18
17																											17
16																											16
15																											15
14																											14
13																											13
12																											12
11																											11
10																											10
9																											9
8																											8
7																											7
6																											6
5																											5
4																											4
3																											3
2																											2
1																											1
0																											0
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	

46	7	8	9	10	10	10	10	10	9	9	8	8	8	8	9	9	9	10	10	11	12	16	20
47	48	49	12	12	12	11	11	11	11	11	11	10	10	11	12	12	13	14	14	15	18	28	25
			28	29	30	30	30	31	31	31	31	32	32	32	32	32	31	31	30	30	28	27	
			30	31	32	32	32	32	33	33	33	33	34	34	33	33	33	32	32	31	31	29	

that a type editor can add dots or take them away in order to create a particular illusion or effect, thus simulating the features of the type face he seeks to emulate.

The coding we have illustrated for the letter "b" would represent starting and stopping points for every portion of every stroke. To store digitized information for a single character by this means, or a similar means called "run length encoding"—which sets forth the number of contiguous black or white cells—usually requires several hundred memory locations (even for a ten-point master). To bring in a single font to store within the memory of the typesetter might require 20,000 "bytes" of information. As we shall see, larger font masters require a great deal more storage.

Electronic sizing.—Once the digitized character is stored it can be used to generate a range of character sizes and shapes. First the sizing itself:

If a ten-point master is stored at a resolution of 720 strokes per inch, it is possible to use *the same master* to provide image sizes up to, perhaps 15 or 16 point characters. This is done by lengthening the strokes and spacing them farther apart. In order to compensate for the greater distance between strokes the dot size may be enlarged somewhat, by increasing the voltage going to the electron gun and causing it to emit a more intense bombardment of electrons. However, the "resolution" of the character has been impaired. A 50% enlargement of the same digital image will reduce the strokes per inch from 720 to 540, just as a 100% enlargement would reduce the number of strokes per inch (not per *character*) from 720 to 360. On the other hand, if the character size were reduced so that the same number of strokes occupied half as much space (say, a character intended for a five-point setting), the stroking resolution would be increased to 1440 lines per inch. There would be the same number of strokes per character but each character would take up half the space. Under these circumstances, even though it is possible to reduce the dot size somewhat, and decrease the intensity of the spot, at some point there would be too much dot overlap and the character would tend to become fuzzy. In general—depending, of course, upon the chosen master-size resolution, the size of the spot and the extent, if any, of character magnification—satisfactory type can be produced by enlarging the image as much as 40 to 60%, and by reducing it as much as 20 to 40%. Thus, a ten-point master could produce acceptable type within a range of, say, from six to 16 points. A 20-point master could produce acceptable type within a range of 16 to 32 points, and a 40-point master could produce good type from perhaps 30 to 64 points. An 80-point master could generate satisfactory images varying in size from perhaps 48 to 128 points. It should be noted that the larger the master size, the greater the number of point sizes it can encompass. There are ten point sizes between 6 and 16, but there are 80 point sizes between 48 and 128!

Consequently, to produce a full range of sizes it is necessary, generally speaking, to store perhaps four different master sizes (and these may be designed to have different characteristics as well, so that they more closely resemble their hot metal counterparts). And for every increase in master size the digital font storage requirements are likewise increased. Since there would be twice as many

strokes in a 20-point master as in a 10-point master, there would be twice as much digital information.

The RCA VideoComp.—The first such "digital" third-generation CRT typesetter was the VideoComp, or, perhaps more exactly, the Hell Digiset, from which the VideoComp was derived. (However, at the same time, a company called Alphanumeric was developing the APS-2.)

Dr. -Ing. Rudolf Hell was a manufacturer of electronic components and devices in Kiel, West Germany. In 1964 he was approached by telephone company interests to ascertain whether he could produce a device that would compose telephone directory galleys on 70mm film. This led to a discussion with other potential customers, and in these explorations the concept of the Digiset emerged —specifically, the product known as the Digiset 50T1, subsequently sold in the United States as the VideoComp 820.

Dr. Hell made his plans known in a speech at an international industry exhibition in 1965, and at that time a laboratory model had been built which set type on 70mm. film. Presentations were made, slides were shown, and in February 1966 a prototype of the 50T1 was shown at the Hannover fair. Just shortly before this, Dr. Hell, in a visit to the States, described the project to RCA at about the time that RCA's newly-formed Graphic Systems Division was getting off the ground. RCA was itself researching the same type of product.

An arrangement was therefore made by RCA to market the Digitset 50T1 as the VideoComp 820. Shortly thereafter RCA began to press for a more sophisticated version, and the 50T2 came into being. This product was marketed in the U.S. as either the VideoComp 830 or 840, or later, as the series-800 VideoComp, with various options. Hell supplied only the back end of the 50T2, and it continued to produce its own hardwired front end for the European market, but RCA used a small computer (the "1600") to serve as a front-end controller for the U.S. market.

Hell delivered a total of 89 Digisets to RCA from 1967 through 1971. These consisted either of 820's or the output end of the 50T2's. At one point during RCA's venture into the typesetter business, there were possibly about 46 VideoComps in the field. Information International, Incorporated (known as III), subsequently took over the VideoComp project when RCA decided to abandon it. III continued to service and to sell the 800-series machines and then developed a 570. In the meantime, Hell went its own way, and has manufactured and marketed a series of its own: the 40T1, the 40T2, and the 40T3. In the interest of brevity we shall discuss only the basic principles underlying the VideoComp products, with particular emphasis upon the features which seem to us significant to indicate the march of technology in the typesetting area.

The VideoComp 820.—The first machine to be introduced was quite limited in the measure to which it could set. The typesetting aperture, when reduced by a sizing lens to approximately three-fourths of its original dimensions, permitted a

line 32.5 picas in length by 56 points high. Size ranges from 4 to 24 points (except for 11 points) could be set, with no half-point sizes. Size range I provided basic sizing of 4, 5, 6, 7, 8, 10 and 12 points, with stroking resolution of 300 to 900 per inch. Size range II offered 8, 10, 12, 14, 16, 18, 20 and 24 points, again with a stroking resolution of 300 to 900 lines per inch. Horizontal size of character could be determined independently of vertical size, but both had to be expressed in terms of the sizes available for the size range of the character in question. Thus an 80-point SR II character could be composed as 10 points in set size, but not as 7 or 9 points since there is no 7- or 9-point available in SR II.

Characters could be obliqued electronically by command, with the normal obliquing angle 15 degrees. The machine would "lead" in a forward direction only.

Optimum performance from both the 820 and the computer used to drive it could be achieved only if all "fonts" required would fit in memory at one time. It was possible to fit four fonts of SR I, or two fonts of SR II (or two of I and one of II).

In the typical 820 installation, fonts would be stored on disc as part of a *computer composition* program. When the magnetic tape to drive the VideoComp was prepared the first fonts required for the job would be pulled off the disc pack and written onto the front of the tape. Thereafter, the input tape would contain text interspersed with font information for such font changes as were required. Reloading the memory of the VideoComp would take a little over one second of time on *both* the computer and on the typesetting machine. Consequently the throughput setting speed would be profoundly influenced by the number of times new fonts would have to be read into memory. Optimum output speeds assuming no font changes could run as high as 600 characters per second (about 500 12-pica lines per minute of five-point type) down to 150 characters per second (about 650 lines per minute) of 24- to 28-point type.

The VideoComp 820 was sold, with magnetic tape reader, for $236,500 in the late 1960's.

RCA VideoComp 830.—The VideoComp 830 used a finer resolution CRT, enlarging the image (2:1) rather than reducing it. This made it possible to set a measure 68 picas wide by 96 points tall, with a one-half pica x 35-point extension on each end; it brought the practical line length to 70 picas—more than double the 820's, and wide enough to set a five-column telephone directory page. Five master font size ranges were offered: the first from 4 to 8 points; the second from 8 to 16 points; the third from 16 to 32 points, the fourth from 32 to 64 points, and the fifth from 64 to 96 points. Different resolution options were possible, but the most commonly used provided for a range between 450 and 900 lines per inch. Size range for SR I included 4½, 5, 5½, 6, 6½, 7 and 8; in SR II: 8, 9, 10, 11, 12, 13, 14 and 16; in SR III, 16, 18, 20, 22, 24, 26, 28 and 32; in SR IV, 32, 36, 40, 44, 48, 52 and 64; and in SR V, 64, 72, 80, 88, and 96. Characters could be set in any combination of point and set size *within the size range.* Leading was offered only in a forward direction, but in increments of 1/32nd point, taking into consideration base line deflection as well as leading "pure and simple."

A micro-image feature was an optional attachment to set 35 mm. microfilm.

Speed on the 830 depended upon the nature of the fonts, but with "Type I" fonts (of superior resolution) setting speeds would typically exceed 1,000 characters per second. Delays were encountered in font loading, as on the 820, but greater flexibility was offered with respect to font storage so that font subsets could be loaded into memory rather than entire fonts.

The purchase price of the 830 was $337,975. An "840" model helped to solve the font storage problem by providing a disc on line to the typesetter's 1600 minicomputer so that fonts no longer had to be loaded along with the text stream either on line from the computer or via magnetic tape.

The Autologic APS-2, the APS-3 and the IBM 2680.—The Alphanumeric Photocomposition System (APS) was developed by Alphanumeric, Incorporated, a Long Island-based company which has now ceased to exist except as a "corporate shell." The original APS-2 CRT typesetter was developed by Alphanumeric for its own use in what was intended to be a phototypesetting service bureau. The machine was redesigned as the APS-3, and was also marketed by IBM as the 2680, beginning in the year 1969. Like most of the other CRT devices, the 2680 did not use the entire face of the tube, but rather a narrow horizontal "aperture" across the middle of the tube. But to maximize setting speed the 2680 composed type on the CRT *uphill* at the same rate that the film advanced. Since film advance was constant (except when it was interrupted due to lack of synchronization), it was necessary to provide a program for the display controller —which operated in a dedicated mode for this purpose prior to typesetting—to calculate the setting speed in relation to the rate of film advance, and to adjust the latter to the former.

The principal difference in its method of character selection from the Video-Comp was that Alphanumeric used a separate master character for *each type size*, rather than electronically enlarging or reducing "master" characters. The normal resolution was 800 lines per inch. Size ranges from 4 to 18 points were offered (with no half-point sizes) and fonts were stored on a disc or drum for relatively fast access. Alphanumeric secured a patent on a method of font compression[1] so that fonts stored on disc could be recorded in a compact manner. The key to this method of font compression was to describe only the manner in which a succeeding stroke differed from the one which went before. A typical 10-point character thus might require only about 80 bytes of information. Hence more fonts could be stored in the memory of the typesetter at one time.

1. Alphanumeric's "Schwartz" patent covered the description of digitized characters by recording starting and stopping points in a manner such as run-length encoding. Their "Manber" patent covered a technique for reducing the amount of information which needed to be stored.

The 2680 offered a maximum line length of 50 picas with a maximum point size of 18 points. Dot size was 1¼ mil with lens magnification from CRT to film plane of 2:1. In the normal mode, with 800 line-per-inch resolution, optimum setting speeds averaged about 2,000 characters-per-second at eight points.

Perhaps twenty 2680 machines were manufactured (to be sold at $385,000 each). A half dozen found their way into Alphanumeric service centers. Most of

the others—if sold at all—were sold abroad, principally to Japan. The APS-3/2680 is primarily of interest because it was the predecessor of the APS-4 and the APS-5—machines which we shall discuss shortly.

The Fototronic CRT.—The Harris Corporation (then Harris-Intertype) entered the CRT market in 1968 when it introduced the Fototronic CRT. A basic machine cost $350,000, with a 512-character font storage memory on a revolving drum. A drum capable of storing 1024 typographic characters raised the machine price to $385,000. A total of some 18 machines were manufactured and sold in the years between 1969 and 1973. Two features were particularly interesting. First, characters were not created by vertical stroking—possibly to get around the Alphanumeric patent(?). Instead, they were "painted" by more or less random movements of the CRT writing beam. Each character consisted of a series of "patches" of dots, and each patch had a common base line reference point. Three, four, or more patches would comprise the entire character. When the character was called for, the typesetter could access a certain track on the revolving drum—with a fixed read head for each track on the drum. That read head would pull off the first available "patch" or segment of the character and continue to read in these segments and to paint them on the output device until the entire character had been created. In general, depending upon the character's size and the number of dots it comprised, one character, in segments, could be stored at least twice on a track in order to minimize access time.

Again, characters were stored for distinct size ranges, and could be sized, obliqued and expanded or condensed electronically. The normal resolution for the master fonts was 720 lines per inch. The lens magnification factor (from CRT to film plane) was 1:1, making for a relatively sharp image, even with a dot size of 2.5 mils.

The machine could set type in sizes from 3 to 28 points in one-half point increments. Characters could be expanded or condensed in one-half point increments over the entire range. Electronic obliquing was also provided at a 20° angle. The normal setting aperture was 51 picas wide by 60 points high, and a "dynamic leading" feature permitted the setting of type "up hill" while the film was being advanced in order to enhance the setting speed. Forward or reverse leading in increments of one-tenth point was also provided, making the Fototronic CRT the most flexible machine, capable of setting the highest quality type, of any machine available on the market prior to the early 1970's. In addition, a film carriage movement was offered which would actually permit the composition of a page as wide as 69 picas (51 picas with an extra 18 picas of line length afforded by the carriage movement).[1]

1. RCA also made one CRT typesetter with a shifting film carriage. It was called the 822X and was sold to Time, Inc. for full-page composition of *Time* pages at R.R. Donnelley.

The Autologic APS-4 and APS-5.—In late 1970, Photon announced that it would market a new CRT phototypesetter to be developed by Autologic, Inc., then a subsidiary of Alphanumeric. The specifications quoted for the machine sounded completely competitive with the VideoComp and the Fototronic CRT but the price, on the other hand, was to be competitive with the Linotron 505. That machine reached the market as the Photon 7000 in 1971, was marketed

both in the U.S. and abroad by Photon as the "7000" and then, later, only abroad, since Autologic took over its own domestic marketing program, calling the device the APS-4. Ultimately, of course, Autologic also made alternate marketing arrangements for overseas sales as well.

Alphanumeric ran out of money trying to operate typesetting service bureaus. A small company it had acquired in California, called Autologic, was able to survive on its own, and inherited the Alphanumeric patents. The engineering work which led to the APS-4 was done by Autologic itself, but there are certain similarities between the APS-3 (2680), and more especially the APS-2, and the APS-4. Like most other CRT typesetters, type is generated a line at a time across the face of a high-resolution cathode ray tube and focused onto moving photographic film or paper. Fonts are digitally coded and are stored on a moving-head disc. A font change is effected simply by calling the appropriate digital coding from disc into the memory of the typesetter's minicomputer controller. Unlike the Alphanumeric machines which required a separate master for each size, the APS-4 adopted the sizing conventions of the VideoComp, but offered many more combinations. In fact, type can be sized in one-tenth point increments so that it is possible to set a character, say, 8.7 points high by 9.3 "set" wide, or 63.8 points high by 52.9 points wide. In addition, obliquing is possible to the user's specifications, which may be changed upon command, and may be implemented as a left or right slant in increments of one-tenth of a degree, from - 45° to + 45°.

The APS-4 uses a typesetting "window" 4¼" wide by one-half inch tall on the face of a five-inch tube. This is generally enlarged 2.2 times to provide a 57-pica aperture, but it is also available with 2.66 magnification to obtain a 70-pica aperture. The APS-4 sold for $125,000 with 16K of memory and a 500 character-per-second paper tape reader. With magnetic tape input and 32K words of memory the price was $146,000. A 100-pica machine was subsequently offered, consisting of two overlapping output tubes, costing $210,000 with magnetic tape input.

The Autologic APS-5.—In the fall of 1975 Autologic began making shipments of a new CRT typesetter priced at under $80,000. It involved a significant redesign of the APS-4, although the two machines are quite similar in basic concepts. The principal differences are that the optical bed in the "5" is mounted vertically rather than horizontally, helping to make the "5" more compact than the "4". The "5" used analog circuitry for correction of distortion across the tube, while the "4" used a digital technique. The "5" is available as a 57-pica machine, using a 2.2 enlarging lens or 70 picas using a 2.6 lens.

A "5U" version could be upgraded to a 100-pica machine. The APS-5/100 uses a 10" tube, rather than a 5" one with a 2X enlarging lens, to set a full 100-pica width.

The APS-5 is offered with a non-removable 2.5 million character disc with optional 10 million and 100 million character storage discs. It offers only one oblique angle rather than 900—namely 12.5 degrees.

Four font masters are required to cover the entire size range: On the 57-pica machine a ten-point master covers 5 through 12.5 points; the 20-point master covers 10 through 25 points, the 40-point master covers 20 through 50 points, and the 80-point master covers 40 through 80 points.

The output writing dot is approximately 1.4 mils at the master size. Stroking spacing at master size is 720 lines per inch and characters are also divided into 720-per-inch segments in the vertical direction. As type is scaled, stroking resolution may vary from 1440 lines per inch down to 600 lines per inch. Because font master size ranges overlap, it is never necessary to use a resolution of less than 720 lines per inch up to 80 point type.

Output speed is limited to 500 cps when reading from paper tape, but when running on line to a host computer or from magnetic tape, input speed will be approximately 2,000 newspaper lines per minute.

It is interesting to consider—given Autologic's font compaction techniques—how many fonts can be stored in memory at one time, and how many can be stored on one 2.5 million byte disc. (Bear in mind that additional or larger discs can also be configured.) The following table indicates the number of characters which will fit into the core memory if the memory is 16 K or 32K, and would thus be accessible without any time loss:

		16K	*32K*
Techno Medium	Size range 1	341	1,024
	2	222	668
	3	107	321
	4	57	173
Times Roman	Size range 1	263	789
	2	160	481
	3	80	240
	4	52	157

In those instances in which an entire font will not fit in core at one time (*e.g.*, a large-sized master font with a 16K machine memory), the font will automatically be segmented and loaded into memory in pieces as needed.

As to disc storage, if every font is required in all four master sizes, at least 30 100-character fonts may be stored on one 2.5 million character disc. But since some fonts are used for text only and some for display, many more text fonts can be stored—in text sizes they take up much less room.

A double-disc configuration would store at least 35 additional fonts in all four size ranges. If only Size Range 1 masters were stored, a double disc system would accommodate at least 600 different fonts.

The MGD Metro-set.—In the year 1973 the Information Products Division of MGD (Miehle-Goss-Dexter division of North American Rockwell) began deliveries of a CRT typesetter called the Metro-set. Several models were introduced. The original Metro-set was priced at $98,500 (and was subsequently raised to $120,000, and then slashed again to the original price but with five free fonts thrown in). Later it was reduced to $92,000—again with five free fonts—and the Metro-set/2 was announced at $79,900 to compete head-on with the APS-5 and the Harris 7400.[1] The Metro-set/plus was also released, resembling the original Metro-set but with a number of options as standard equipment, including hyphenation and justification logic with the 4K of extra memory required to run it, a program to accept justified wire service tapes, and ten newspaper fonts—all for $98,500.

1. See page 147.

It may be seen that the decision on the part of Autologic to produce the APS-5, and to price it so competitively, led other companies to modify their products and/or prices. This trend has continued, especially with the introduction of Mergenthaler's Linotron 202. Consequently, it is now possible to purchase a digital CRT at a price within the same range as some of the second-generation typesetters.

The Metro-set differed from other CRT typesetters primarily in one very interesting respect. While characters were generated by vertical strokes, as with the VideoComp and APS models, these strokes were not stored as digital starting and stopping points. What was stored on disc was a *description of the character's outline,* made up of subdescriptors of outline segments. When these outlines were called from disc to the typesetter, specially-configured hardware studied the outline of the character (much as if it were scanning it) and then generated a stroking pattern for the desired size. This meant that, unlike most other digital CRT typesetters, MGD would set type at any size with the same number of strokes per inch, just as is the case with the CRT typesetters using photographic masters for font storage. Three resolution modes were offered: one of 744 lines per inch and a second at 1488 lines per inch. In addition, a low resolution proof mode was provided, with a stroke density of 496 lines per inch. In all modes, the "resolution" in the vertical direction remained a constant 1,024 dots per inch.

To set type from a given font master, the Metro-set would call up from disc the digital coding which would describe that font and load it directly into one of the two 4K memory banks in the stroking processor. Metro-set fonts could contain up to 126 characters and there were almost no fonts which could be completely contained in 4K of core memory. Depending upon the complexity of the font, between 50 and 100 character masters would fit into memory at one time. (The average is about 80 characters.) So, a portion of the memory bank was reserved for so-called "transient" characters. The Metro-set would "look ahead" into the input text stream in order to load the coding for these characters from disc into core before they were required. If a particular text called for heavy use of a wide variety of transient characters, the machine could slow down while

waiting for these characters to be pulled off disc, but this almost never happened. The Metro-set would also look ahead in the input text stream for font changes and load the coding for the next font to be used into the second bank of font memory before it would be needed.

The Metro-set was the first CRT typesetter to utilize look-ahead logic to minimize the time loss in making font changes. Since the Metro-set was introduced, Harris and Mergenthaler have also worked out look-ahead logic on a character-by-character basis for the 7400 and the Linotron 606, and Compugraphic has built look-ahead logic for font changes into the Videosetter Universal.

Font scaling.—Unlike prior digital CRT typesetters, the Metro-set used one character master to cover the entire range of sizes to be set. In essence, the font stroking processor "scanned out" the characters to be set as if it were always going to set a 72-point character, then "threw away" strokes to reduce the image on the tube down to the point size specified by input command. In practice, the Metro-set had very few restrictions on type sizing except those imposed by the input command language for the machine. The maximum type height which could be set was 72 points, and the width was something on the order of three inches. This latter feature lent itself to the generation of logotypes as characters. *(See below.)*

The Metro-set used a large 11″-diameter output tube with a fiber optics face plate. (Remember that the Compugraphic Videosetter also uses a fiber optics face plate!) The tube was suspended vertically from the machine and the output film or paper is pressed flat against the face of the tube. In other words, no magnification or reduction by lenses could be achieved, and the image was always set at the same size at which it was generated on the tube. The typesetting aperture was 56 picas wide by one inch tall.

Another interesting feature was that the Metro-set would set as many lines of type as would fit within the one-inch aperture before advancing the film. It did not attempt to set type while the film was moving. However, output film or paper could be moved in a forward direction, or reversed, subsequent to each filling of the aperture with type.

The use of the concept of storing characters as digital outlines or portions thereof may have originated from a desire to avoid patents on other methods of font storage. If so, it illustrates how the effort to get around patent limitations may well contribute a valid, and conceivably superior, alternative solution.

As with Compugraphic, the use of the fiber optics face plate presumably would inhibit the development of a typesetter capable of setting a 100-pica measure for full page-width newspaper composition.

MGD offered a logotype scanner, plus an auxiliary disc unit, for on-line storage of up to 250 logotypes, for $32,900. Other graphics options included the ability to set reverse type (white on black) and to generate screens and tints.[1]

1. At this writing, it appears that the entire Metro-set product line has been discontinued by Rockwell.

The Harris Fototronic 7400.—Harris made a study of the prospects for CRT typesetters if sold at a lower price than their $300,000 CRT. They realized that the market for a machine in the $300,000 range was limited, but they also concluded that only a relatively small market existed for a lower-priced version. Consequently, for several years they put their plans to develop such a typesetter "on the back burner." On the other hand, precisely during this period, Autologic demonstrated that there was indeed a substantial market for such devices in the under-$150,000 price range, and in the meantime technology improved to permit the development of CRT typesetters at such a price—devices at least as good as—and in most cases substantially better than—the $300,000 machines which were introduced in the late '60s. In 1974 Harris decided to re-enter the CRT market, and in June, 1975 it introduced the 7400, for delivery in 1976.

The Fototronic 7400 generates its characters from a store of digitized fonts, using the vertical stroking method, with some degree of font compaction. Fonts are stored on an individual character basis, on one or more 2.5 million byte disc drives. The typesetter's control program reads ahead in the input text stream and calls the digital representations of the characters into memory from disc ahead of the time they are required for setting purposes. Fonts are stored at different master sizes, and type may be scaled up and down from the master size by squeezing or expanding the dot matrix pattern on the output tube. Four font masters are generally required: a 7-point master covers sizes from 5 through 10 points; a 14-point master covers 10 through 20 points; a 28-point master covers 20 through 40 points, and a 56-point master reaches from 40 through 96 points.

Type can be sized in increments of 1/10 point in both horizontal and vertical directions, from 5 points through 96. Three right obliquing angles are offered, to be selected by input command: 9, 12 or 15 degrees.

Stroke spacing is 723 dots per inch at master size. As type is scaled, resolution will vary from 1000 dots-per-inch down to 500 dots-per-inch (up to 80-point type) and above 80 points, dot spacing will be less than 500 dots-per-inch, dropping down to 420 dots-per-inch for 96-point type.

Characters are stroked on a 10″ diagonal CRT which is also positioned facing upward in the cabinet. The image is deflected longitudinally by a high-quality mirror, and focused through a 1.3X lens onto the photographic medium. Tube spot size for the 68-pica machine runs from 1½ to 2 mils, which means that dot size on the film plane runs from 2 mils to about 2½. Harris aims for a dot overlap from 30 to 40% for ideal resolution.

Twenty full-range fonts can normally be stored on one 2.5 million byte disc; a second disc drive can be added. Speed for straight matter setting is estimated at 1,000 newspaper lines per minute—somewhat slower than some competitive models—but in mixing, due to the look-ahead feature the machine fares very well.

Harris has also released a 100-pica version—the 7600—which makes use of the same-sized tube, but with a different magnification factor.

The VideoComp 500.—The latest models in the VideoComp line are the 500 and (as of the fall of 1976) the 70-pica VideoComp 570. The "500" makes use of the output tube and optical bed of the older VideoComp 800 and the Hell Digiset 50T2, but with a new front-end controller built around III's own minicomputer. On the other hand, the 570 represents a completely re-engineered and much more compact "back-end" as well.

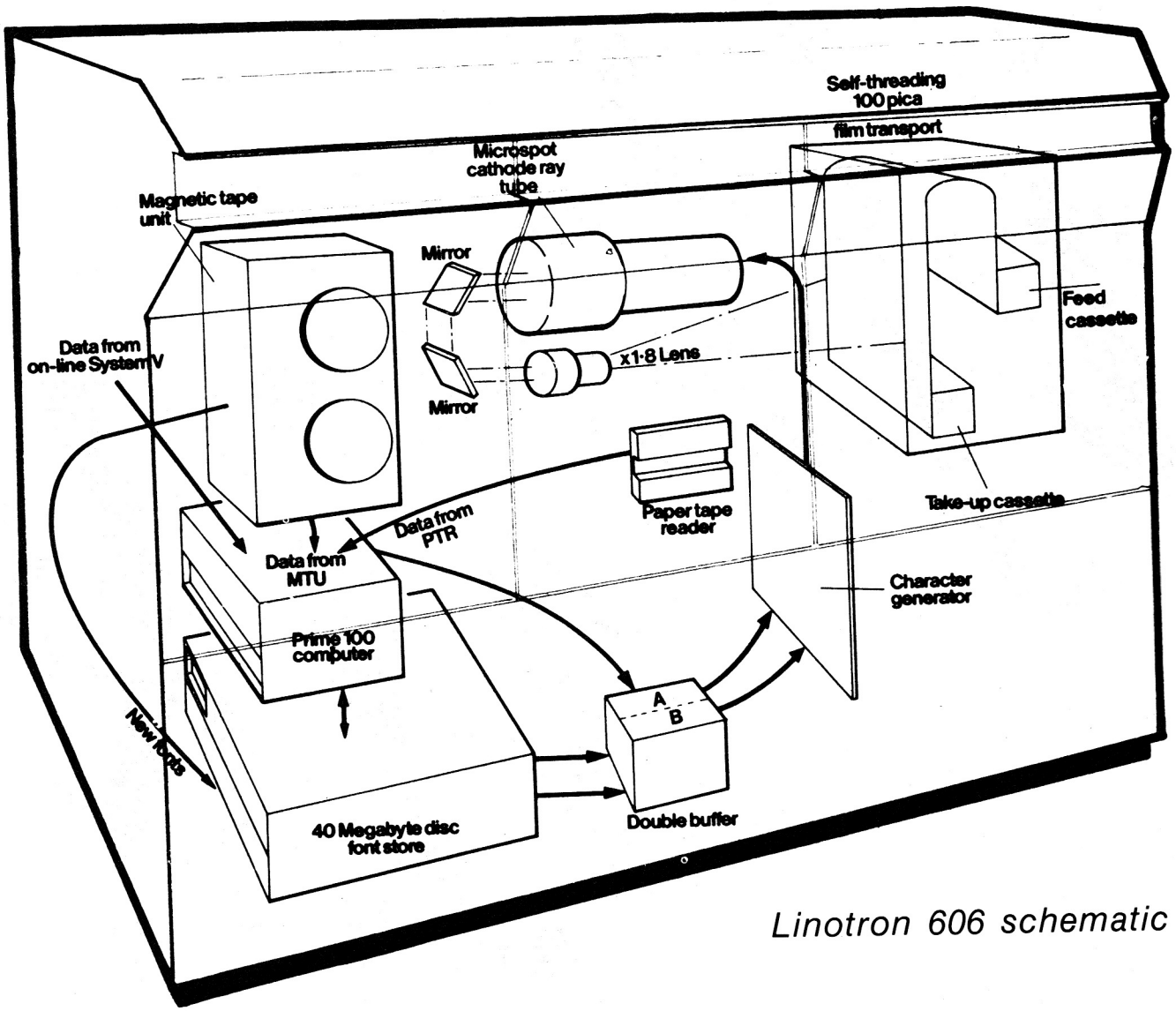
The standard "500" (and the 570) operate in a line-at-a-time mode, similar to most of the other CRT typesetters, but a "full-face" option is also available, which most users prefer, especially since the "500"-series machines do not offer reverse leading. Multi-column composition can be produced simply by relocating the writing beam and the device also offers superior vertical rule generating capability. The full face 570 permits the composition of a page 54 by 66 picas without moving the film. Operating in an aperture mode, it can set 72 points wide by 70 picas long, a line at a time. (A 90-pica version is optional.) Size range 1 fonts cover 4 to 8 points, size range 2 fonts will cover 4 to 18 points, and size range 3 fonts will reach from 16 to 32 points. Size range 4 fonts, from 16 to 72 points, are only available on machines with the full face option. Type can be sized in increments of 1/10 point, both horizontally and vertically, which seems to be becoming the standard for the better grade of CRT machines. Any angle between 6 and 15 degrees, in one degree slants, is offered for obliquing.

The horizontal resolution for size range 1 fonts varies from 900 to 450 strokes per inch, and for size ranges 2 through 4 it will vary from 1800 lines to 400 at the top of the size range. Vertical resolution remains at 2,160 dots per inch.

Standard memory will hold the equivalent of 32 disc sectors of font storage. One Times Roman font would take up 8 disc sectors in SR 1, 12 sectors in SR 2, 20 sectors in SR 3, and 36 sectors in SR 4. Since a standard disc has 4200 sectors, at least 50 fonts could be stored on line on the standard 2.5 million byte disc. An optional 20-million byte disc would hold nine times as many fonts.

The Mergenthaler 606.—The 606 was Mergenthaler's first digital CRT typesetter. It was introduced in 1976. Fonts are stored on an individual character basis on a high-speed 40-million byte disc drive. The digital representation of the character is read off the disc *every time the character is to be set.* Nevertheless, due to look-ahead logic and stroke compaction, the 606 has the fastest claimed output speed of any CRT typesetter presently on the market. As with the Autologic and Harris machines, fonts are digitized as stroking patterns and stored as different master sizes. Type may be scaled up or down from these sizes and the user apparently will be able to specify the master sizes he desires.

The 606 offers a 100-pica line length. (Mergenthaler released a 70-pica, lower-cost simplified version, known as the 404 in 1978.) The setting area of the 606, as enlarged by a sizing lens, is 2½" tall by 100 picas wide. Machines may set more than one line of type before advancing the film, and will continue to set type while advancing the film when setting in smaller sizes. Type can be sized in increments of one-half point, horizontally and vertically, from 4½ to 128 points (with accents



Linotron 606 schematic

and long descenders), or to 160 points for normal descenders, and "caps only" to 256 points! Stroke spacing at the master size is 650 lines per inch, varying from a high of 1,625 strokes (at 40% of master size) to a low of 371 strokes at 1.75 times master size. The 404, on the other hand, sets from four to 96 points, in half point increments. Setting speed is rated at 1500 newspaper lines per minute (as contrasted with the 606's 3,000).

The Linotron 202.—Also in 1978, Mergenthaler Linotype introduced the Linotron 202. Like the 606 and the 404, it too was designed and manufactured by Linotype-Paul, in Cheltenham, England. But the 202 is a radically different machine.

In the first place, like the Metro-set, characters are stored as outline segments. But, unlike the Metro-set, all such segments consist of straight lines,



used as minutely as necessary in an attempt to form graceful outline curves. Then, these stored outlines, brought into computer memory from disc, are stroked by a software program (the Metro-set used hardware to generate stroking patterns) to provide a constant resolution of 975 lines per inch. The same font masters are used for all sizes from 4½ to 72 points.

The speed of the 202 is considerably less than the 606 or even the 404. It is approximately 450 newspaper lines per minute—about the same as the Videosetter, and from one-third to one-sixth the speed of an APS-5, VideoComp 570, or Linotron 606 and 404.

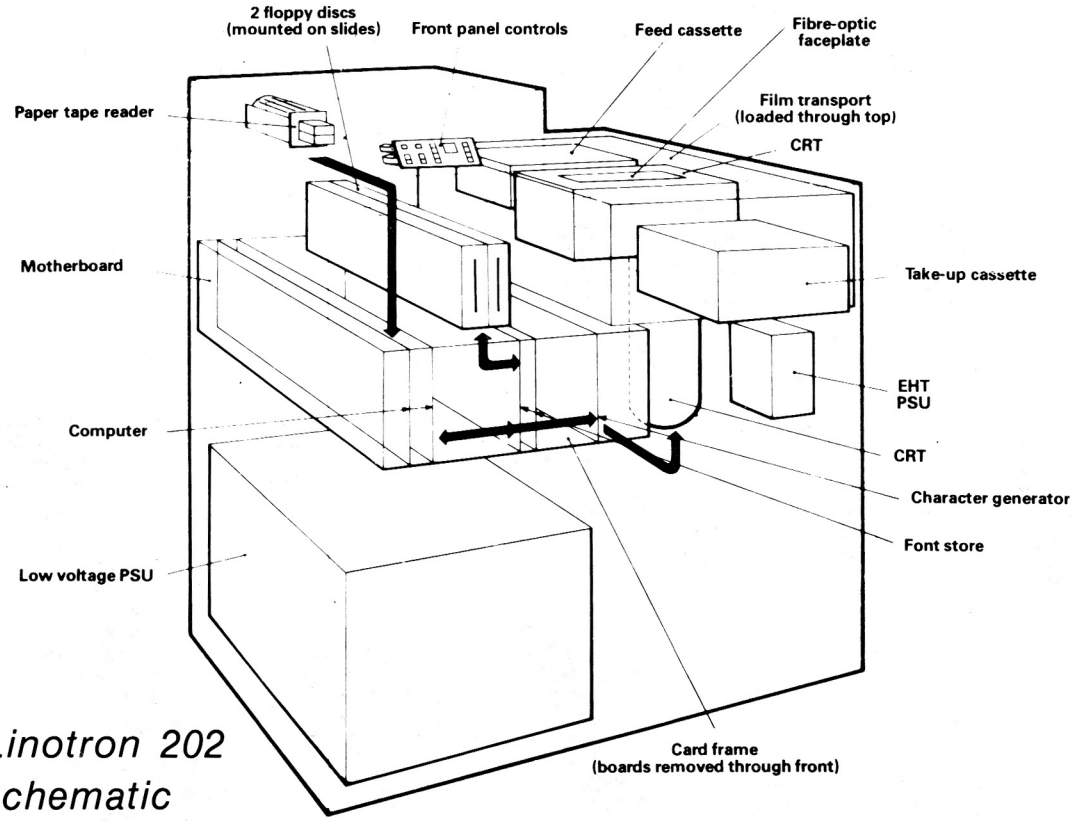
A second distinction is that—again like the Metro-set—the 202 uses a fiber optics face plate and has no lens. The maximum line length of the machine is 48 picas. The price of the 202 is competitive with the Compugraphic Videosetter Universal.



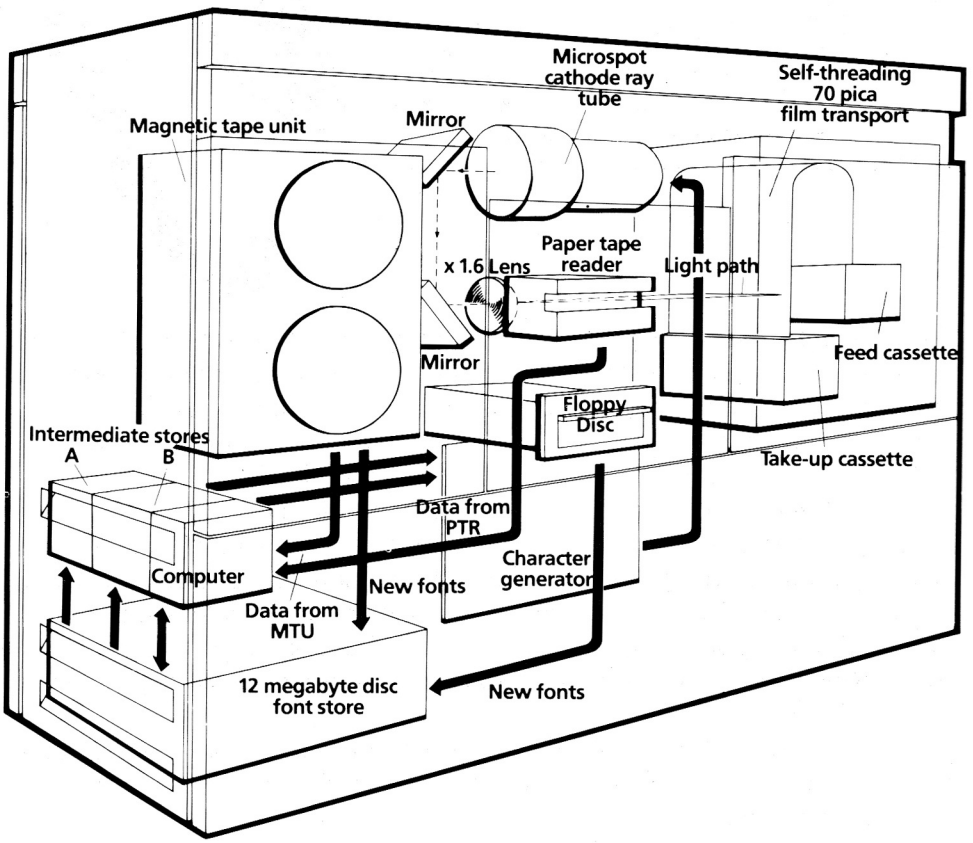
Seventy-two point characters set on the 202, the 404, the 606 and the Mergenthaler V-I-P, enlarged to 400% of original size. In contrast to the V-I-P, you can discern the effect of vertical stroking on the 404 and 606. On the 202, the enlargement shows straight line segments which make up the character outline. Under more normal conditions, the straight line segments are only visible in the largest point sizes, and only then if you look very closely at the typeset output.

Other CRT machines.—In 1977-78 Alphatype introduced its CRS machine. This, oddly enough, is a slow-speed CRT typesetter, offering an unusually compact character resolution. In some respects it is similar to the Linotron 505 and 303, since escapement is provided by mechanical movement—in this case the movement of a travelling lens which steps across the film plane as each vertical stroke of the character is generated. The stroke size (generated on a very large scale by means of a very inexpensive TV tube) is reduced by a lens in the ratio of 30:1, thus affording the incredible "resolution" or stroke spacing of 5300 strokes per inch. Adjacent strokes overlap by approximately one-third. The spot size on the film plane is approximately one-third of a mil. Stated output speed is 50 newspaper lines per minute in sizes not larger than twelve points.

The device as first released sets its type on a sheet of lith film, rather than on a continuous roll of film or paper. It is expected that a cassette version will be introduced subsequently.



Linotron 202 schematic



Linotron 404 schematic

The user will maintain a library of fonts on individual floppy discs, with a number of master sizes on the same disc. He will pull from the various discs the digitized masters (both face and *size*) that he requires for a given job, writing them, in turn, onto one or more floppies which will be accessed by the typesetter.

One would expect that subsequent versions of this machine might seek to store more fonts on line, perhaps on a rigid disc.

More to come?—It will be evident to the reader that vendors have shown no lack of imagination in designing new third-generation typesetters. There is no reason to expect that other similar machines—with different combinations of features—will not be introduced. Prices are coming down perceptibly at the same time that the technology is improving.

"Fourth Generation" Technology

All of the typesetters which we have described as "third generation" are CRT machines. You will recall that some of them store their type images photographically and some digitally. Some of the machines require a moving part to sweep the strokes across the film plane. Most do not. Some (the VideoComp 500/570 and certain models of the Hell Digiset) offer a full-face capability, while the others set type within an aperture of the tube. Some of the latter move the paper or film continuously across the face of the tube while others either move the paper continuously or set up-hill while the paper is advancing (dynamic leading). For the most part those machines that compose within an aperture offer reverse leading, to permit the setting of multi-columns, while this is not necessary on a full-face machine, where the beam can be directed to the area where the next block of text is to be positioned.

Some machines magnify or reduce the image between the CRT tube and the film plane. Others use a fiber optics face plate which transfers the image same-size from CRT tube to film.

Some machines also offer a microfilm or microfiche option, so that the image generated on the face of the tube can be written out as full pages in the very minute size which micro-imaging devices require. And some CRT typesetters make it possible for the user to digitize and reproduce not only type but line drawings or halftones. *(See Chapter 22.)*

But we have not exhausted the possibilities in the design of typesetters, by any means. It is possible for a laser platemaker to be a typesetter. It is possible for an office copying machine to generate its own type. It is possible, at least in theory, for an electronic camera to set type. Most of these devices make use of laser technology, and most of them function in a "raster-scan" mode. For present purposes all we need to know about the laser is that it is capable of generating a very fine and very intense spot of light. It is possible to interrupt this light source so that the light emissions can be pulsed or timed to coincide with some other event. It is also possible to deflect the light from a mirrored surface, and to rotate

that mirror so that the light beam will sweep a given area. Within certain limits it is also possible to deflect the light beam in a vertical direction. This, indeed, was what Dymo Graphic Systems did with their DLC-1000 "fourth generation" typesetter—a device which was briefly introduced to the marketplace and then discontinued. What the DLC-1000 did was to deflect the laser beam vertically by means of an acoustic-optical coupler, thus generating strokes which were "laid down" by a rotating mirror. One reason that the product was discontinued was that the supplier of the acoustic-optical deflection device was a relatively small manufacturer with an uncertain future. Another reason, perhaps, was that it was necessary to store a digital representation of each type character in every desired point size.

On the other hand, the Monotype Lasercomp uses a raster scan technology. The laser beam strikes a rotating multi-faceted polygon with mirror-like surfaces. This sprays a very intense blob of light, which is focused down to a very fine spot and directed through a field-flattener lens (to correct for deflection distortion) so that a series of one mil spots are sprayed horizontally across a film plane that could be as wide as 100 picas. The film continuously advances and a new series of spots are sprayed across the surface of the photographic film or paper. Letter shapes thus emerge, since the dots which make up the individual characters are stored in memory. A line of type—in fact a tiny horizontal slice of an entire newspaper page—is thus generated from the formatted text held in the typesetter's buffer memory. Here, again, a separate master is required for each desired point size.

As composition programs, including page make-up, become more and more sophisticated, and as computer memories become larger and cheaper, it becomes increasingly possible to hold in memory a "bit map" of all the tiny dots or spots that a laser would be called to lay down during one sweep across the page. The polygon mirror system moves at a constant speed. It is the timing of the laser light pulses that determines where the dots are laid down.

While this technology seems very complex, it is likely that this method of setting type will be used increasingly, and there are similarities between the Lasercomp, the ECRM Autokon camera, and the EOCOM laser plate-maker.

Summary

Initially, third-generation typesetters were extremely expensive. Moreover, since they were "slave" machines, they could only be driven by paper or magnetic tape which had been formatted through a computer program. In those days the only computers available for this purpose were at least as expensive as the typesetters themselves—if not moreso. Consequently, very few firms could afford to purchase or lease systems of this kind, and the only way they could be cost-justified was on the basis of a very large volume of typesetting. Programs to enable the computers to output to typesetters were not very efficient, nor were they very reliable. It was especially difficult to provide perfect "input" for these computer programs, and the larger systems proved to be viable only for the

processing of large "data bases" such as telephone directories, parts and price lists, the airline guide, and cumulated abstracts and indexes, such as the *Index Medicus* of the Medlars project of the Department of Health, Education and Welfare.

Attention therefore shifted to minicomputer solutions, and less expensive typesetters of the second-generation variety. It was clear, moreover, that with the tremendous setting speeds of these third-generation devices, very few firms could manage to keep them busy.

Furthermore, the newspaper industry, which might have provided a major market—not so much because of the volume of composition required, but because of the amount which has to be produced during a very short period of time—was not then ready or able to take advantage of these typesetting devices for a variety of reasons. Perhaps the major reason was that most of these papers were printed by letterpress, and there was no inexpensive way to go from photographic material to a letterpress plate—especially multiple plates. There were also union obstacles in the way of management in introducing the new technology, and these obstacles were often more severe in the newspaper industry than they were in the commercial sector.

However, over a period of perhaps ten years, the situation shifted quite dramatically. In the mid-1970s, newspapers constituted the principal market for CRT typesetters, and some emphasis came to be directed toward providing machines capable of setting entire pages—even, perhaps, including halftones and line art. Later in the decade, commercial firms that had not been able to afford CRT typesetters found that at the new lower prices they could no longer ignore them. They also found that the quality of output had significantly improved so that it was good enough for most applications. And the ability to keep more fonts—"on-line"—or include one's entire font library—greatly simplified the scheduling of work and significantly reduced typesetter down time.

Only the CRT typesetters, and especially the digital machines, seem to offer the capability to the newspaper to store all type fonts "on-line" so that there need be no concern for the "dressing" of the typesetter. Any available work can flow to the typesetter at any time. And the speed is available to set a "dump" of the entire classified ad section of a metropolitan newspaper in a few minutes' time, so that last-minute ads, coming to the newspaper by telephone, could be keyed, processed, incorporated by computer program into their proper sequence, and then output in vertically-justified galleys or even in pages. Only with the development of such total solutions was it possible for newspapers to move away from earlier technologies, where ads were set as received, and manually-positioned blocks of type were shuffled and reshuffled into galleys and pages.

CRT typesetters offered other conveniences for the commercial or non-newspaper industry. Proofreading anything less than type itself proved to be difficult and expensive. Consequently, a typesetter fast enough to generate output several times, as corrections and revisions were made, became more and more

attractive. Moreover, logotypes (special symbols used to designate trade names, *etc.*) can be digitized and reproduced. Some CRT typesetters can even typeset digitized halftones.

At one time it was no doubt true that the quality of type produced on such machines was not equal to that of hot metal linecasters or of at least certain second-generation photosetting devices. This is—generally speaking—no longer the case today. And the typographic flexibility—in terms of sizing and character repertoire, as well as font diversity, and the "on-lineness" of the operation—provides persuasive reason for all who are occupied with the tasks of setting type to consider the "third-generation" solution. Attention is beginning to focus, moreover, on "fourth-generation" solutions using raster-scan technology.

Review Questions

1. What is the difference between a "second-" and a "third-" generation typesetter?
2. What are the two general categories of third-generation typesetters?
3. How does a cathode ray tube (CRT) work?
4. Describe the principles which underline the Linotron 505. How does the 505 differ from the Videosetter?
5. Of what value is the ability to raise characters above (or set them below) the base line?
6. Why do typesetter manufacturers provide the capability of electronically condensing or expanding type? How is this accomplished?
7. In what principal ways does the 303 differ from the 505?
8. What technological changes made it possible for the Videosetter to achieve such low costs?
9. What problems are posed when the "full face" of a CRT is used for character generation rather than to use an aperture or "window" of the tube?
10. Why can a CRT typesetter set a wider measure in its "aperture" than it can elsewhere on the face of the tube? Why does the height of the aperture limit the line's setting length? Do the 505 and the 303 compose type in an "aperture"?
11. When a digital CRT such as the VideoComp or the APS-4 enlarges the point and set size generated from the stored master image, does the "resolution" go up or down?
12. Why (or when) would "look-ahead logic" increase setting speed?
13. Some digital CRT typesetters do not store fonts as strokes. What are these machines, and how do they store them?
14. Why do large font masters take up more memory or disc storage than smaller ones?
15. How does the font storage of the Metro-set differ from that of the APS-4 or 5, or the VideoComps?
16. How do the Metro-set and the Linotron 202 construct characters as opposed to other digital CRT machines? What does this mean in terms of character "resolution"?
17. Using a piece of graph paper, experiment with the plotting of a few sample characters, limiting all "dots" to squares within the grid outline provided by the graph paper. How large a grid (in terms of strokes or graph-paper squares per em square) do you require, in your opinion, to create acceptable characters which resemble normal type characters in their appearance?

10

Basic Computer Concepts

COMPUTERS are a relatively new phenomenon in modern society, and there are few inventions—throughout the history of the world—that have had greater impact. But then, the science of mathematics—or, more precisely, of numbers—is itself of more recent origin than many of us may have imagined. Positional notation—such as the decimal system—apparently was first developed by the Babylonians about 3000 B.C., but was subsequently lost to posterity. The Mayans, around 200 A.D., developed what is said to have been a comparable concept, but it, too, had no impact upon "our" civilization. The Hindus, on the other hand, made substantial contributions to algebra during the period from 500 to 1200 A.D. But it was the Arabs, in the ninth century, who developed the concept of decimal notation, and it was not until the twelfth century that roman numerals began to be superseded by Arabic numbers in Europe. We are told that the Arabic system was in general use for scientific purposes by 1400, although "no Arabic numerals are to be found in English parish registers or Manor Court rolls before the sixteenth century."[1] The multiplication table was still a mystery to most educated people even in the seventeenth century!

1. See the discussion in *Electronic Computers*, by S.H. Hollingdale and G.C. Toothill, Penguin Books, 1965, especially p. 25.

However, mechanical calculating devices (as distinguished from the ancient abacus) came into being in the seventeenth century and until the middle of the twentieth century gears were employed to provide a method of "carrying" from one row of numbers to another. The development of a "setting" counter and a "result" counter made the mechanical process of addition possible.

Blaise Pascal (1623-1662), the French philosopher and mathematician, is generally given credit for the development of the first mechanical calculating machine. This was essentially an adding and subtracting tool. Multiplication could be carried forward only as repeated steps in addition. Leibnitz (1646-1716), the German mathematician, contributed to the ability to mechanize multiplication. For every addition, a counter clocked the result, and division was derived similarly from the process of subtraction.

But writers reporting about modern computers generally start their history with Charles Babbage (1791-1871). It was in 1812 that he conceived the notion of computing and printing tables of mathematical functions through the construc-

tion of a "Difference Engine." Unfortunately, his machine was never completed, although several such devices were subsequently built. Babbage turned his attention to the idea of building an "Analytical Engine" which was to be a universal computer rather than a mere device for the solution of polynomials. In other words, Babbage's first concept was a "special-purpose" computer, and his second was to be a "general-purpose" instrument.

These were the days in which inventiveness revolved largely around the capacity to devise intricate mechanical "machines." Babbage's task was therefore to contrive ways of storing information in the form of gear movements. Each gear, with ten teeth, could in turn drive a second gear that would advance only one position when the first gear completed its entire rotation, and so on through each succeeding gear sequence. But to avoid the problem of setting the gears initially by hand, Babbage thought he might be able to use a system something like the one that Joseph Marie Jacquard (1752-1834) had developed for the weaving industry.

Jacquard had contrived a system of activating rods which would, in turn, lift warp threads so that the weaver's shuttle could pass underneath them. Each of a set of cards would be brought into contact with a set of rods in such fachion that, in those instances in which the card contained no holes, the rods would be pushed into an alternative position.

Babbage conceived of the idea of using somewhat similar cards to serve two purposes: one, to handle the *input* data, and two, to *store* the results—*output* or *interim output*.

Babbage continued to work on his project until his death in 1871. The problem was not, however, with his conception. It was "merely" that the device as conceived was too complex mechanically to provide sufficient accuracy and reliability. It was Augusta Ada, Countess of Lovelace (1815-1852), the only child of Lord and Lady Byron—a philosopher and mathematician in her own right—who undertook to create a chronicle of the Babbage project and who contributed substantially to its conceptual implications.

Our attention next focuses on Herman Hollerith, an employee of the U.S. Bureau of the Census, who took the Jacquard punched-card concept and applied it to the recording of such Census data as could be answered by "yes" or "no" questions, or by grouping data into categories. By 1890 he had several machines in operation. It is believed, as we have already suggested, that his concepts influenced Tolbert Lanston in the development of the Monotype system.[1]

1. See page 44.

The tab card era

By the year 1910, Hollerith tab card systems were beginning to be used quite widely, and, until the advent of the modern digital computers, "tabulating" systems handled the bulk of the world's mechanized "data processing" applications. But it is significant to note that, by that time the tabulating process, with its use of punched cards, had turned to electrical circuits—unlike the "rods" em-

ployed by Jacquard, the gears of Babbage, or the columns of compressed air utilized by Lanston. When a punched card was fed into one of the several machines developed to perform various types of calculating and related tasks, electrical contacts were made. The reading of a "hole" in a column was accomplished by an electrical brush coming into contact with a metal roller.

Among the tasks to be performed was the "sorting" of alphabetic lists. Generally this was accomplished by reading information from only one of the 80 columns at one time. Thirteen alternative vertical holes are available in each column, and thirteen pockets could be provided for sorting purposes. A plug board could alter the electrical circuits so that not all thirteen alternatives needed to be selected, and some "de-selecting" could also occur.

Once the first column was examined and cards assigned to bins in accordance with the value of the figure in the first column, a similar processing run could take place for all the cards which had been sorted into the first bin, in order to subsort these and to arrange them in a secondary sequence. Then all of the cards with common letters or digits in the first two columns could be sorted with respect to the third column, *etc.*

Many other functions could be performed by tabulating equipment, such as addition, subtraction, multiplication, and even printing (*e.g.*, checks or tables of results), and these tasks could be performed at what seemed to be relatively high speeds, since such cards could be read at the rate of several hundred per minute, and various types of calculations or functions could be carried forward at comparable speeds. Cards thus provided the initial input, and results could also be recorded automatically on similar cards. Moreover, information could be printed on a tabulating or printing unit which works very much like an all upper-case typewriter.

And so, between the period of World War I and World War II, the tabulating machine came into its own. It was no doubt World War II which provided the impetus for the first truly electronic calculator. This invention is generally credited to J.P. Eckert and J.W. Mauchly, of the Moore School of Electrical Engineering, at the University of Pennsylvania, although a British professor, Alan Turing, an American zoologist, Norbert Wiener, and Howard H. Aiken, all did pioneering work which culminated in the 1938 model of the Automatic Sequence Controlled Calculator of the Mathematical Institute of Harvard University.

The Eckert-Mauchly *ENIAC*—Electronic Numerical Integrator and Calculator—was intended to assist in the computation of missile trajectories. Eniac differed from its predecessors in that it performed all of its functions by means of electronics, except for the tasks of reading or punching cards. Eniac contained 18,000 vacuum tubes! It was not a binary, but rather a decimal, machine. A special circuit was developed which kept track of pulses and "carried" a digit when the tenth pulse was received.

However, Eniac was a "special-purpose" computer. Each "program" had to be loaded into the machine by wiring the required functions.

Eniac was in operation by 1945, and in 1946 John von Neumann came to the fore in presenting more clearly the analogy between the performance of *logical* functions and those which had to do with mathematical or arithmetic calculations. It quickly became apparent that binary systems were both more reliable and easier to design than decimal systems, and were better able to perform logical tasks. Finally, during the 1950s, and particularly during the 1960s, cheaper, faster, and more dependable circuits were constructed which no longer required the use of vacuum tubes, but could use transistors in the first place—and, somewhat later, silicon chip techology.

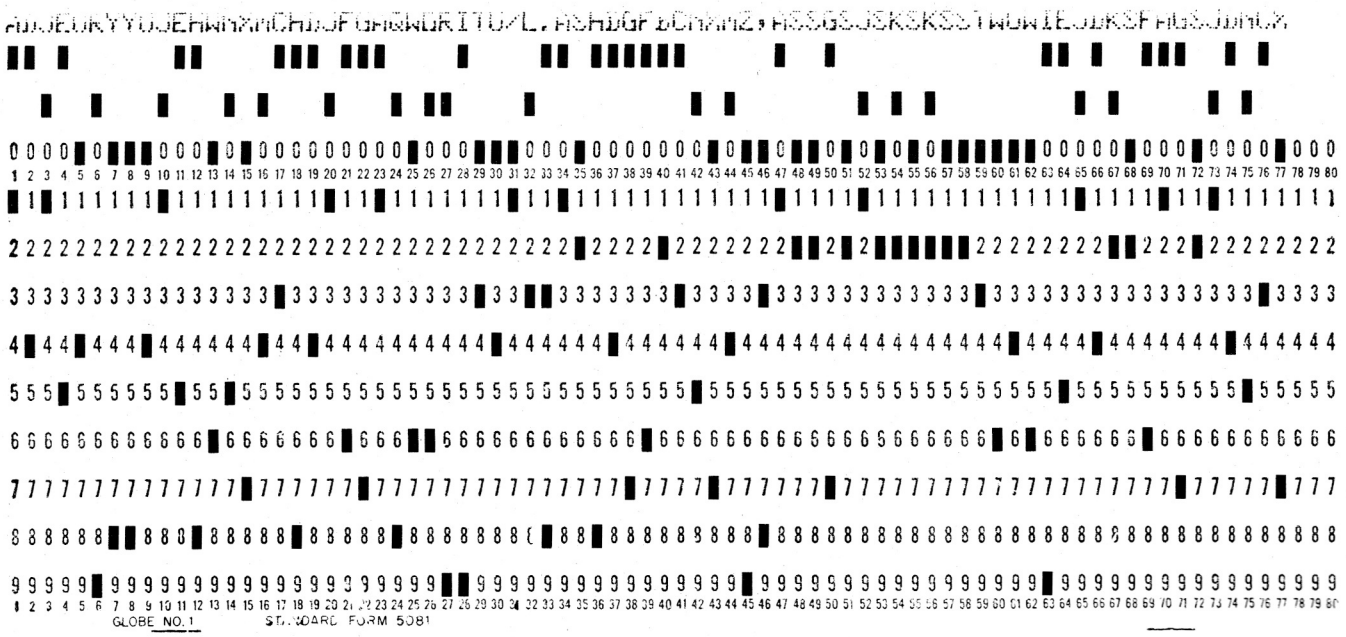
Two problems—apart from cost—plagued the vacuum tube technology: one was the problem of reliability. It was simply impossible to rely upon 18,000 or more of such tubes to work as they were supposed to function, all at the same time. The second problem was the amount of heat generated, which had disastrous effects upon other electrical components. Transistors, happily, were developed just at the critical time. They generate very little heat, take up little space, last hundreds of times longer than vacuum tubes, and can therefore be configured into more elaborate systems, capable of performing functions much more rapidly. Transistors are "semi-conductor" devices—consisting mainly of a crystalline substance, such as silicon and germanium, into which certain impurities are introduced. Two negative-type semiconductors are generally separated by a positive wafer. These three substances are usually sealed in plastic, with a wire leading from each segment. The wires are connected to an electric circuit, and if the electronic components of the segments are properly designed, a small current flowing between the emitter (one *n*-type element), the base (a *p*-element), and the collector (another *n*-element) produces a current amplification.

The transistor was the product of a joint effort of John Bardeen, Walter H. Brattain, and William Shockley (all of whom later shared a Nobel prize). It was announced by the Bell Laboratories in the year 1948. A transistor has no filament, requires no warm-up, and generates little heat. Transistors can be mounted on small boards and connected together by "wire wrapping" or by "printed circuits." These boards or panels can be mounted vertically and side-by-side within a small area. Individual boards can be easily replaced.

It was not until the mid-'50s and the early '60s that transistor technology took over. In the late 1960s and early 1970s, LSI (large-scale integrated circuits) of MOS (metal oxide silicon) chips introduced another dramatic change, since just as one small circuit board (say, of postcard size) could replace a large cabinet of vacuum tubes, so one chip the size of a pinhead can replace several hundred circuit boards.

The use of Boolean logic

George Boole, English mathematician and logician (1815-1864) was a relatively obscure professor at Queen's College, in Cork, Ireland, and published, in 1854, *An Investigation of the Laws of Thought* as well as some works on cal-



The "IBM" or "Hollerith" card is still widely used for data entry purposes. In its current version it most commonly contains 80 columns, and these cards are prepared by "key punch operators." There is also a verifying key punch, whereby a card which has already been prepared is re-inserted into a machine, and an operator prepares a new card. If there is any lack of correspondence between the "old" card and the keystrokes about to be entered, this condition is flagged so that the operator can determine whether it is she or the former inputter who made the mistake.

The 80 columns of the card may be arbitrarily divided into fields. For example, the first 20 columns could be assigned to enter the last name of a subscriber, the next 10 columns could be used for the first name, and the balance assigned for address and zip code fields. It is possible for the information to require more than one card. It is also possible—indeed, likely—that some information can be described by an entry in only one column of the card, such as a personal or family income bracket classification, or an age grouping.

Twelve positions are available in each vertical column. The ten bottom rows customarily represent numbers, 0 through 9. One punch hole per column is thus used to indicate figures. When a hole is punched in one of the top two rows (or in the zero row)—*in addition* to one of the rows, one through nine, a non-numeric character is customarily indicated. For example, "A" consists of a hole in the top row and a hole in the "1" row. "B" is produced by punching a hole also in the top row, and a hole in the "2" row. The letter "J" is created by placing a hole in the second row as well as in the "1" row. "S" is signalled by the use of a hole in the "zero" row plus a hole in the "2" row.

A total of 41 characters can thus be described, consisting of ten numbers, 26 capital letters, and five miscellaneous characters, some of which serve as control codes, but the period, comma, semi-colon and dollar sign are frequently assigned.

It is possible to use the same card to convey more information by punching more holes in these vertical rows, combining codes much as you would on paper tape, but this is seldom done, even where the input is into a computer system rather than a tab card installation. However, the use of an upper- and lower-case alphabet is sometimes encountered.

culus and differential equations. Boole sought to apply mathematical principles to logical thought and was therefore one of the founders of *symbolic logic*. He developed a set of concepts perhaps more basic than the Aristolelian propositions (*e.g.*, syllogisms) and certainly more valid for the analysis of problems and the pursuit of inductive reasoning processes. Most of the logical propositions that are currently related to the computer's electronic circuitry, and much of the programming logic which utilizes this circuitry, derives from Boole's work, and from his propositions, which can be stated in terms of electronic circuitry design, or as logical statements. Improvements in methodology were contributed by Augustus De Morgan (a contemporary of Boole's), John Stuart Mill, Lord Bertrand Russell, and Alfred North Whitehead.

Logical propositions lie at the heart of the computer's electronic circuitry. To be sure, there are many tasks which are procedural in nature: the "reading in" of data, the "output" of information, and even the performance of arithmetic functions. But there must be an underlying set of logical principles which can determine whether certain things are alike or different. You can't, of course, add apples and oranges, but you can count up the number of pieces of fruit on the table. We can therefore handle information constructively if we are painstakingly aware of the respects in which things are similar and in which they are different from each other. Thus we may find it necessary to talk about "people" (living or dead?), and also males and females, girls and boys (how defined?), as well as persons with other characteristics, such as education, experience, income, attitudes, and innumerable other variables—all of which relate, somehow, to classes which serve to include or to exclude for whatever analytical purposes we may have in mind.

Such logical tasks may be performed quite simply by electronic circuitry. It is not within the scope of this book to provide an exhaustive account of the principles of computer circuitry any more than it is to provide a dissertation on symbolic logic, but simply to give the reader some feeling for the nature of the problems and concepts that are relevant to a general understanding of the computer world.

Some electronic circuits

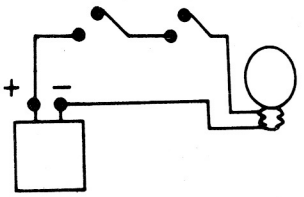
What are the functions which computers are able to perform? Their tasks and capabilities are many and varied, and yet the essential processes are made up of a relatively small number of logical components. Basically, the ingredients consist of "and-gates," "or-gates," "negators," and "flip-flops."

These are, in substance, "Boolean" concepts which have electronic circuitry counterparts. Let us consider several of them:

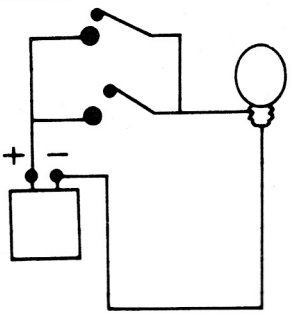
A "gate" is what its name implies: some things enter and some things leave. But not everything which seeks to enter may be permitted to pass through. In the case of an "and-gate" only those things can leave which meet *all* the relevant

tests. There must be more than one such test. There may be two. There may be more than two.

Let us consider the "and-gate" which requires the presence of two conditions. We may say that only those things are "true" which satisfy both conditions. Visualize a light bulb, then, which is connected to a battery with two switches wired in series. The light will only turn on (truth!) if both switches are closed. If either switch remains open the light cannot be illuminated. Hence "a" *and* "b" must both be true.

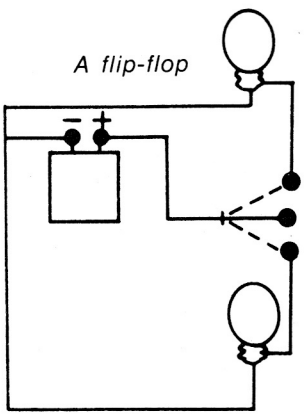


On the other hand, the "or-gate" could be like a light bulb in a circuit so wired that *either* of two switches, *or both*, will turn on the light. You can pass through the gate if you are either male or female. Both males and females may pass through the gate.

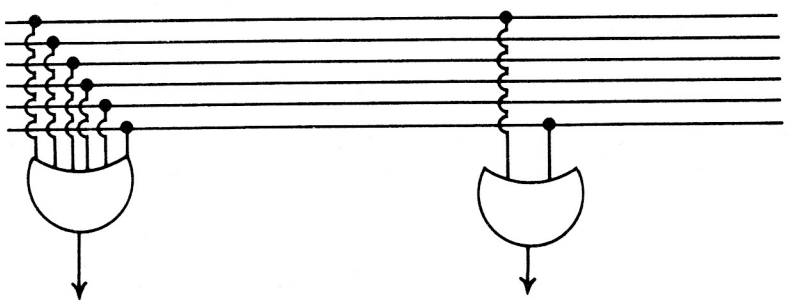


A "negator," on the other hand, has the simple property of turning whatever exists into its opposite, assuming that there is a true opposite—and the opposite may even be everything that isn't one specific thing. If a negative charge of electricity is sensed, it will be converted into a positive charge instead.

A "flip-flop" has the property of changing the state so that *either* one condition is true *or* an alternative obtains. Whereas a "negator" transforms one thing into another, a "flip-flop" selects one thing *rather than* the other, and what it selects is not necessarily the opposite. It simply leads us down a different path.



Without belaboring the point, it should be clear that quite complex logical decisions can be made electronically. For example, if we want to indicate a six-level code as one of 64 possible conditions to be recognized from a TTS tape, the logical diagram would look something like the following:



We have shown only two of the logical circuits necessary to pick out certain combinations of six bits. The first diagram, on the left, presumes an "and-gate" which would require the presence of inputs from all six circuits (0 through 5). The second diagram, on the right, shows an "and" gate which need only detect input from "0" and "5." Obviously, a similar gating arrangement must be provided for every possible combination (63 in this instance).

The structure of a computer

Now let us consider the structure of the computer itself. A computer may be said to be made up of five components:

1. One or more input devices.
2. Communicating channels.
3. A "main frame" or "central processing unit" (CPU).
4. A "memory."
5. One or more output devices.

Increasingly, however, we will encounter other "peripherals"—that is, units which are added onto the computer system and somehow interfaced to the computer in order to perform additional functions or to provide additional resources. A supplementary store of data may need to be consulted. A work area may need to be provided where tentative results may be stored for later reference. Methods of communicating may be desired for users who are located remotely.

Later on we shall discuss input and output at greater length. For the moment let us merely suggest that input can come (among other sources) from punched cards, punched paper tape, or from scanning and sensing devices.

Output can be directed toward a printer (usually called an on-line printer), or a console typewriter. Punched cards and magnetic tape can also serve for output purposes. Output sometimes consists merely of signals that activate some other mechanism (such as a punch or milling machine in the case of "process control" devices).

Communicating channels.—At this point we wish to direct our attention to the communicating channels which connect the main frame or CPU to the input and output devices, and often link memory to the CPU itself. These channels or *buses* are also used to connect other peripheral devices which serve as a temporary store for data, such as magnetic tape drives, or discs.

A channel not only *transmits* the flow of data, but also regulates and controls this flow so that it is synchronized with other operations. When a peripheral device is ready to transmit, the channel controller will be signalled. It will then accept the data or a manageable slice of it, often in a fashion which permits the virtually simultaneous transmission of other data in an overlapping manner. Information which is received as input may be accepted in an "asynchronous" mode (which means that it comes more or less without warning and certainly not on any pre-arranged or "clocked" schedule), or it may be "synchronous" (which means that the information is expected, and is "clocked" so that its processing corresponds to the operating speed or "cycle time" of the computer or of some of its components).

There are often two kinds of channels: multiplexor channels and selector channels. A multiplexor is used for slow peripheral equipment and breaks the channel down into what could be thought of as a number of sub-channels. Data

may be transmitted in minute elements, in some order or sequence, from different devices, and are assigned work or storage areas which in effect permit the assembling and processing of data from one unique source, relating to one unique task, without regard to the fact that other information has been intermingled during the course of transmission. The parallel example in the telephone industry is the ability to transmit many different conversations over the same wire at the same time without any user being bothered by, or even aware of, the fact that others are sharing the same line. Information which is received on a multiplexor channel may also be received in a "bit serial" fashion, rather than "parallel"—which means that it is received a "bit" at a time and, before processing, must be reconstructed into an array of bits which are considered simultaneously.

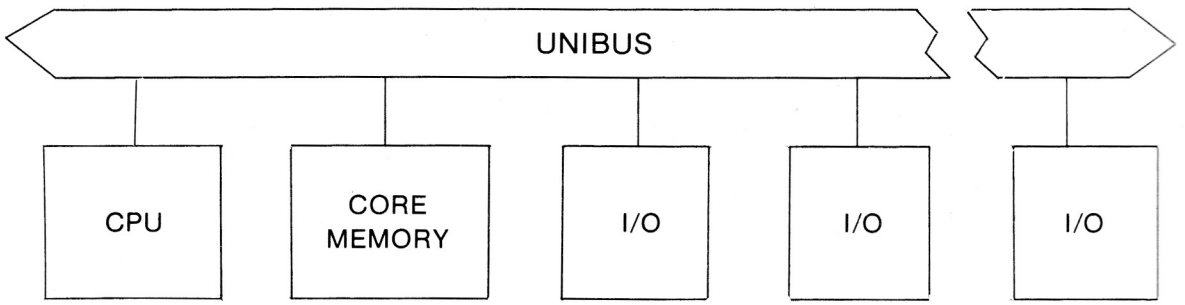
A selector channel, on the other hand, generally conveys information from only one source at a time, in what is called a "burst" mode—which is a very rapid infusion of data which fully occupies the capacity of that high-speed channel for the period of time of the particular transmission.

To make it possible for channels to be used in these two ways, and especially to be capable of accepting several different kinds of assignments, such as to transmit data from input device to CPU on one occasion and from CPU to output device on another, there are controllers which operate in conjunction with the CPU to accept and process data in multiplexor and burst modes. Controllers vastly enhance the efficency of the main frame, since the latter can be commanded to work on tasks already in hand rather than to wait until new data may be transmitted. Similarly, the processing of new information need not be delayed until the old has been disposed of (as output) since the CPU can continue to perform available work. In other words, the main frame is not so likely to be "input bound" or "output bound" and it can spend its time doing computations, rather than having to wait for input or output devices to perform their tasks.

Channels and their controllers also provide the main frame with certain other services: they will interrupt an operation to inform the CPU that an error condition exists, or that the input has been terminated. They will also perform certain "storage protect" functions which prohibit the inadvertent destruction of stored information by preventing data from being written on top of useful tables, program areas, etc. "Read only" features exist which permit data to be picked up from a given location, but nothing new may be written to that location.

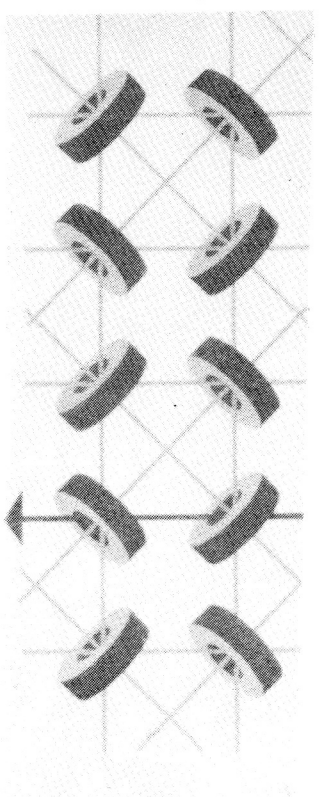
Another common type of "computer architecture" involves the use of the "Unibus" structure which is found in Digital Equipment Corporation's PDP-11 minicomputers. In this case all computer system components and peripherals connect to and communicate with each other on a single high-speed bus known as the *Unibus*. With bidirectional and asynchronous communications on the Unibus, devices can send, receive and exchange data independently without processor intervention. Device communications on the Unibus are interlocked. For each command issued by a "master" device, a response signal is received from a "slave," completing the data transfer. Device-to-device communication is independent of the physical bus length and the response times of master and

slave devices. Moreover, interfaces to the Unibus are not time-dependent. Input/output devices are nevertheless given the highest priority by a "bus arbitrator" and may request "bus mastership" and "steal" bus and memory cycles during instruction operations. The processor resumes operation immediately after the memory transfer. Multiple devices can operate simultaneously at maximum direct memory access (DMA) rates by "stealing" bus cycles.



PDP-11 system simplified block diagram

It may be seen, then, that there are various ways in which the main frame may be related to its peripherals, and that the task of controlling many diverse operations is quite complex.



The main frame or CPU

The main frame generally has these elements:

1. A control unit.
2. A storage unit.
3. An arithmetic and logic unit.
4. A set of registers.

The control unit.—This unit interfaces with the channels and their controllers, and it also coordinates the various functions within the main frame itself. Control is partly "hardware" (wired instructions built into the computer), partly "software" (programs), and it can also be partly "firmware"—*i.e.*, programmed instructions which change the logic paths of the machine as they might otherwise be hardwired. It will not be necessary to our present purposes to discuss the intricacies of *control*. But it should be obvious that the ability to execute commands and to keep track of data are essential to the functioning of the computer.

Storage.—Data coming into the main frame needs to be put somewhere until it is processed, and then arranged in a store somewhere until it can be presented as output. Data (including the programs themselves) will be pigeonholed into "locations" which have "addresses" so that the information can be found when needed.

Data elements are stored in *memory* (or on disc, drum or tape) in the form of "bits," "bytes," and "words." A "bit" is the presence or absence of an event which can generate an electric pulse—whether the event is a hole in a paper tape, or a positively or negatively charged minuscule doughnut-shaped piece of ferrite. A bit is a switch, in a sense, which is always in an *on* or *off* position, and it can be interrogated in order to find out which condition governs at the moment. Since we can therefore only consider two conditions, we can describe them as either "0" or "1"—"no" or "yes." Two bits taken together can describe four conditions:

00
01
10
11

In like manner, three bits can describe eight conditions:

000	100
001	101
010	110
011	111

Five bits can describe 32, six bits can describe 64, seven bits can describe 128, and eight bits can descirbe 256 different conditions.

Six-, seven- or eight-bit "bytes" are used most frequently to describe characters—especially in typesetting. Some experts prefer to use the term "byte" to describe a string of eight bits only. The manner in which characters are stored in memory is a function of the size of the "word" or string of bits hitched together at one address and generally manipulated at one time (moved from one place to another, for example). In the IBM 360 and 370-series computers, four bytes make up one word, and two bytes are considered a "half word."

For purposes of doing arithmetic, a four-bit byte may be sufficient. If we wish to deal only with decimal numbers we have need only for 0 through 9, plus a few control characters which should stand for functions like "end of message" or "end of record." But for the treatment of alphanumeric data (numbers plus letters), six or more bits in a byte are more convenient. If we are not concerned with typographic detail—such as upper and lower case characters—and if our character set or repertoire need not be extensive, a six-bit byte will be more than ample, because we would only need:

26 letters
10 numbers
10 punctuation marks (approximately), and
6 or so control codes

52

Therefore all of the most commonly-desired characters could be described with six bits, and a computer printout device (such as an on-line high-speed printer) with a print chain of 64 characters is quite adequate.

By the use of a precedence code for upshift characters and another to signal a return to lower-case (and assigning other symbols to the number codes in upper-case), a six-bit code structure—as in TTS—is sufficient, although an eight-bit byte offers greater latitude.

Using a six-bit code structure, and a computer with a six-bit "word" we can visualize computer storage as consisting of memory which looks something like this:

Location	*Bit Configuration*
0	0 0 0 1 1 0
1	0 0 0 1 0 0
2	0 0 0 1 0 1
3	0 0 0 1 1 1

This could represent the number 6457, beginning at memory location "0."

If we wanted to store the word "word" beginning in location 0, that portion of memory might look like this: (In this instance we are using an arbitrary code structure—TTS—to describe the letters we wish to store.)

Location	*Bit Configuration*	*Character*
0	0 1 1 0 0 1	W
1	0 0 0 0 1 1	O
2	0 0 1 0 1 0	R
3	0 1 0 0 1 0	D

The concept of parity.—Often, for purposes of assuring machine and transmission accuracy, an additional bit is provided for in memory, and it may also be present on the paper tape or magnetic tape which transmits information to and from the computer. The additional bit serves as a check to see if the device which writes out or reads the codes is functioning properly. This is a "parity check." Wiring and logic are added to the computer to cause it to count the number of "1" bits in a byte. If this number is odd, an additional "1" bit will be added to make it even (or *vice versa* if it is supposed to be odd.)

Parity Bit	*Other Bits in Byte*	*Without Parity*
1	0 0 0 0 0 1	odd
0	1 0 0 0 0 1	even
1	1 1 0 0 1 0	odd
0	1 1 0 0 1 1	even

A parity check means that the memory must always be configured to accommodate one additional bit for purposes of error checking, even though the parity bit is ignored for other purposes.

Other methods of parity checking may be utilized, such as a "longitudinal parity check" whereby all of the bits in a certain vertical row (for a given block of

data) are added and checked to determine whether they are odd or even. Then a check bit is written at the end of each block, corresponding to the finding for that particular row.

What is a convenient byte size?

In a later chapter we will discuss special problems of typographic representation, storage, and information transfer, which arise from our typesetting requirements. At this point we need say no more about the manner in which characters are stored and described other than to indicate that the trend is toward the use of eight-bit bytes, providing 256 distinct code combinations. The code structure which eight bits affords is often called "hexadecimal" although this term more properly describes a method of notation based upon the eight-bit byte coding structure. Hexadecimal means that the numbering system most convenient to this code structure is not based upon a "set" of ten, but rather upon a "set" of sixteen. A numerical "set" is a concept which relates to the number of units which can be described by a single digit (occupying only one row) when used for counting purposes. "Sets" are "new math" concepts well understood by children but not so easily grasped by adults who learned their arithmetic some time ago. Perhaps we should digress briefly to talk about sets and particularly the hexadecimal and octal sets. But first let us consider the concept of "binary" notation.

Binary.—A binary set is made up of only two numbers: 0 and 1, and yet it is possible to count to infinity with such a system. The problem, nevertheless, remains one of describing the numbers you are counting. In pure binary arithmetic the only numbers you have to work with are zero, one, ten, eleven, one hundred, one hundred and one, one hundred and ten, one hundred and eleven, one thousand, one thousand and one, and so forth. It might be difficult to follow the statement that I have 101 fingers on one hand and that the sum of fingers on both hands is 1010, and so it is easier to use binary notation, but to "think" in terms of a more inclusive set, such as decimal, octal, or even hexadecimal. Here we have a string of binary numbers with their decimal equivalents:

Binary	*Decimal*	*Binary*	*Decimal*
0	0	110	6
1	1	111	7
10	2	1000	8
11	3	1001	9
100	4	1010	10
101	5	1011	11

It may be observed that in a binary set:

$1 = 2^0$ or 1
$10 = 2^1$ or 2
$100 = 2^2$ or 4
$1000 = 2^3$ or 8
$10000 = 2^4$ or 16
$100000 = 2^5$ or 32

The binary set is made up of only two numbers, 0 and 1, but it can be used to express decimal concepts. The table we have shown above, setting forth binary and decimal numbers, illustrates this. On the other hand, an "octal" set would consist of only eight numbers, from 0 through seven. Here is the relationship between octal and decimal:

Octal	*Decimal*	*Octal*	*Decimal*
0	0	11	9
1	1	12	10
2	2	13	11
3	3	14	12
4	4	15	13
5	5	16	14
6	6	17	15
7	7	20	16
10	8	21	17

But an octal set can be described in binary numbers as well as in the arabic numbers 0 through 7:

Binary Codes	*Octal*	*Decimal*
000 000	0	0
000 001	1	1
000 010	2	2
000 011	3	3
000 100	4	4
000 101	5	5
000 110	6	6
000 111	7	7
001 000	10	8
001 001	11	9
001 010	12	10

Notice that if the six bits are divided into two parts (and we have so represented them for convenience) it is easy to describe the configuration of bits in octal terms, by describing the first three binary bits as one number and the second three binary bits as a second number—or, more precisely, as a second digit. Hence the octal "10" can also be a decimal "8" or a binary 001000.

But the use of the octal method of describing six-bit bytes (looking at them as if they were two three-bit bytes side by side) is highly convenient. Thus:

000 000 is an octal 00
000 100 is an octal 04
001 001 is an octal 11
010 100 is an octal 24

This is a very convenient arrangement, since we can then create a "table look-up" arrangement which appears like this:

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	8	9	A	B	C	D	E	F
2	G	H	I	J	K	L	M	N
3	O	P	Q	R	S	T	U	V
4	W	X	Y	Z	!	.	+	"
5	%	¢	&	*	(	)	-	—
6	.	,	?	/	‘	’	$	
7								

The vertical row of numbers outside the block represents the first digit of a two-digit octal number. The horizontal row on top of the block represents the second digit. Thus an octal "11" is a "9" and an octal "40" is a "W."

The characters shown in the above grid have been assigned arbitrarily, and not all of the codes have been used up, but it will be noted that we have made our decimal "8" an octal "10" and the letter "T" is an octal "35."

The hexadecimal system

The same principle is involved in the hexadecimal system, except that our number set, instead of consisting of eight digits, 0 through 7, consists of 16, 0 through "F." In other words, the set counts out as 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F.

Decimal	*Hexadecimal*
9	9
10	A
11	B
12	C
13	D
14	E
15	F
16	10
17	11

And so on, until we reach:

25	19
26	1A
27	1B
28	1C
29	1D
30	1E

Then 31 becomes "1F" and 32 is a hexadecimal "20."

When these concepts are understood we can then discuss "binary coded hexadecimal," which looks like this:

0000 is a hexadecimal 0
1001 is a hexadecimal 9
1010 is a hexadecimal A (which is a number)
1011 is a hexadecimal B
1100 is a C
1101 is a D
1110 is an E
1111 is an F

Just as three bits can enable us to count 0 through 7, so four bits can count from 0 through 15, and the number 16, in decimal "set," brings in the second, or left-hand, group of four bits:

0001 0000 is a hexadecimal 10, although a decimal 16.
0001 0001 is a hexadecimal 11 and a decimal 17.
0001 0110 is a hexadecimal 16 (what is its decimal value?)
0001 1111 is a hexadecimal 1F.
0010 0000 is a hexadecimal 20.
0010 1111 is a hexadecimal 2F.
0011 0000 is a hex 30 (decimal 48).
1111 1111 is a hex FF (decimal 255).

It is obvious that the conversion from hexadecimal to decimal is more difficult to "do in your head" than is conversion from octal to decimal. People who work with hex numbers usually have recourse to a table to save the labor of thinking through the relationships. And others prefer to stay with octal. In the latter case they divide the grouping of the eight-bit "byte" into the following array:

00 000 000

Thus we can describe the bit combinations in octal terms, bearing in mind that the left-hand grouping could never equal more than 3:

11 111 111 would give us octal 377, which is a decimal 255, and which of course remains hex "FF" if we regroup the bits into two arrays of four.

Returning to the hexadecimal concept, a translate table can be prepared which shows the two hexadecimal characters as coordinates. *(See top of next page.)* In this case the left-most digit is usually shown at the top, and the right-most digit at the left side.

Here we have used both upper and lower case letters, arabic numbers, and all the punctuation marks and special symbols you will generally find on a typewriter (plus, perhaps, a few more), and we still have not begun to fill up the matrix of characters!

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	g	w	M	:	]										
1	1	h	x	N	;											
2	2	i	y	O	‘											
3	3	j	z	P	’											
4	4	k	A	Q	-											
5	5	l	B	R	—											
6	6	m	C	S	(											
7	7	n	D	T	)											
8	8	o	E	U	$											
9	9	p	F	V	%											
A	a	q	G	W	¢											
B	b	r	H	X	&											
C	c	s	I	Y	*											
D	d	t	J	Z	@											
E	e	u	K	.	#											
F	f	v	L	,	[											

The characters we have shown in the above matrix are again arbitrarily assigned and this assignment does not accord with any code structure in common usage. The reader may be interested to study two different code structures which are commonly employed in information transfer. One is called EBCDIC —pronounced ("Ebsadick")—which stands for "Extended Binary Coded Decimal Interchange Code," and the other is USASCII ("Yousaskey"), or ASCII ("Askey")—which stands for "United States of America Standard Code for Information Interchange." We have chosen to show (on the next page) both EBCDIC and ASCII eight-bit or "hex" codes on one diagram. Displaying them in this fashion does not emphasize sufficiently the number of blank spaces (unused code assignments), although some of them are taken up with information-handling codes rather than information-content codes.

It may be seen, then, that the modern general-purpose computer, being by its very nature a *binary* device, processes data as strings of bits which are organized into bytes and words, and that man has devised several (actually many!) code configurations for the purpose of representing computer instructions and alphanumeric information. Data will be brought into the computer as a string of bits, it will be processed as bits, arrayed in some combination, using programs expressed as bits, and it will be stored and output in the same manner. During the production or processing cycle various translations of bit streams or bytes may take place since the input coding structure, the internal coding conventions and the output coding requirements may all differ.

Shortly we shall return to our discussion of the functions of the CPU, now that we have laid some background for this purpose, but it may be helpful first to consider the *speed* at which the computer functions—especially its central processing unit.

EBCDIC and USASCII Coding

Left half of byte in hexadecimal coding

Right half of byte in hexadecimal coding		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0	0000			Sp	0	Sp @	& P	- 	p					{ 	} 		0
1	0001			 |	1	A	Q	/ a	q	a	j	~ 		A	J		1
2	0010			"	2	B	R	b	r	b	k	s		B	K	S	2
3	0011			#	3	C	S	c	s	c	l	t		C	L	T	3
4	0100			$	4	D	T	d	t	d	m	u		D	M	U	4
5	0101			%	5	E	U	e	u	e	n	v		E	N	V	5
6	0110	LC		&	UC 6	F	V	f	v	f	o	w		F	O	W	6
7	0111	Del		'	7	G	W	g	w	g	p	x		G	P	X	7
8	1000		Can Can	(	8	H	X	h	x	h	q	y		H	Q	Y	8
9	1001			)	9	I	Y	i	\ y	i	r	z		I	R	Z	9
A	1010			*	:	¢ J	! Z	 j	: z								
B	1011			+	;	. K	$ [	, k	# {								
C	1100			,	<	< L	* \	% I	@ 								
D	1101	CR CR		-	=	(M	)]	_ m	' }								
E	1110			,	>	+ N	; 	> n	= ~								
F	1111			/	?	| O	- 	? o	" Del								

In the chart above, characters in EBCDIC are in the top portion of the box. Characters in 8-level ASCII are in the lower portion. Hence, "A" is 11000001 in EBCDIC and 01000001 in ASCII, but lower-case "a" is 10000001 in EBCDIC and 01100001 in ASCII. Sp=Space, LC=Lower Case, Del=Delete, Can=Cancel, UC=Upper Case, CR=Carriage Return.

Computer speeds

One of the most interesting facts about a computer is the speed with which it can process information. We have already suggested that a computer may be limited in its processing speed by slow input or output devices—and this is all the more reason for processing several jobs simultaneously while waiting for input to come in or for output to be disposed of. But the "cycle time" of the computer itself can be extremely fast, and thus even if the basic tasks a computer performs are repetitive and often exceedingly complex, they can be done very quickly and the computer has made possible the performance of very complicated cal-

culations which could not otherwise have been performed even within many lifetimes.

Computer speeds are generally described in terms of "cycle time"—which is the time required to move one "word" of data from one location to another, as from one place in core (memory) to a register, or from a register to another location in memory. Remember that a "word" can be the length of several characters. A 16-bit word machine can simultaneously describe two eight-bit characters. An IBM 370 "word" can contain four eight-bit characters.

Often instructions take up a computer word, and hence cycle time may relate to the length of time it would require a computer to fetch (but not to execute) the instruction.

Only a few years ago we were impressed that computers could execute such a cycle in a certain number of "milliseconds." Thus, a nine-millisecond cycle time would mean that a computer could move a word in .009 seconds. This is a little less than one one-hundredth of a second. Thus a hundred such operations could be performed each second. But now computer speeds have moved into the microsecond range (if not even faster!). Just as a millisecond is a thousandth of a second, a microsecond is a thousandth of a thousandth of a second. So a computer cycle time of six microseconds would be .000006 or 6/1,000,000th of a second. To express this speed another way, such a computer could perform 166,666 operations in a second! And there are computers which operate in the nanosecond range—which is a thousandth of a microsecond.

Suppose you ran one job and only one through a computer that operated at a speed of 800 nanoseconds, and suppose that the input from this job consisted of reading in one character at a time from a paper tape, on a paper tape reader that operated at 1000 characters per second. Of course there would be other tasks to be performed, but if only one cycle were required to bring this information from paper tape to a computer register then the computing power to be used for this operation would require only one out of every 12,500 available cycles. It would be as if you were to spend one second on a task and then wait for the assignment of another task (occupying only one second) three and one-half hours later! If the basic task you were expected to perform were one which required an hour, then your next assignment would be a year and one-half later, and if it took a day, you would only have to perform this duty once during your entire working life span. These examples are suggested to dramatize the problem of computer speeds and computer utilization.

The instruction set

We have established the fact that a computer can perform certain input/output operations (generally called I/O), utilizing its channels and controllers, and that it can also store data in its memory in binary form. But how does the central processing unit manipulate the data? These are the functions which are performed by the use of its arithmetic and logic unit, with the assistance of its

"registers," storage capabilities and controllers. What *are* the arithmetic and logical functions the CPU can perform? These are a function of the machine's *instruction set.*

Different makes and models of computers have different instruction sets depending upon their design characteristics and the applications for which they are intended. An instruction is called an "operation" and it is expressed by an "op" code. The command tells the computer what to do and where to do it, where to get the data and where to put it. Operations may be *declarative,* which defines the element to be worked with, and *imperative,* which then orders that something specific be done with that element. What is to be done and the circumstances under which it is to be performed are set forth in what are called *arguments.* There may be imperative commands of an arithmetic nature (add, subtract, multiply or divide), or of a logical nature (do this if this *and* that are true; do this *until* you have done so a certain number of times or until something else—which is specified—occurs). There are also imperative commands of a manipulative nature—move this right or left, fill it with zeroes, pack or unpack the data, translate it, align the decimal points, *etc.*). Or, read in a block of information from magnetic tape; print out a block of information on a line printer.

Here are some common imperative commands, set forth in a simplified version of a possible instruction set:

Add	Execute	Start
AND	Load	Store
Branch	Move	Subtract
Compare	Multiply	Test
Convert	OR	Translate
Divide	Pack	Unpack
Exclusive OR	Read	Write
	Shift	

These instructions (and others which are more specific) are effected by "machine language" commands contained in an "object program," but they are executed by the arithmetic and logic unit, assisted when necessary by channels and controllers and peripheral devices. The wiring of the unit can permit it to perform certain tasks very fast, as we have already seen. The principle behind binary addition, for example, is that:

$$0 \times 0 = 0$$
$$0 + 1 = 1$$
$$1 + 0 = 1$$
$$1 + 1 = 10$$

Instructions are executed electronically by invoking the use of certain circuits which are able to perform logical functions, in the manner we have already discussed. It is also possible for the computer to be "micro-programmed"—which means that many of the instructions can consist of software subroutines which perform functions similar to those which are "hard-wired," or even more complex

instructions. Furthermore, computers can be programmed to perform multi-processing functions, which means that one program can be interrupted and another can take its place. In this case the contents of the first program's registers will somehow be stored so that when that program is resumed it will have its own data to work with and its own links to indicate what the conditions were when the task was interrupted.

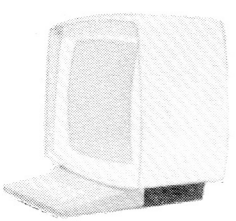
The use of registers

Registers are devices for storing one or more "words" in order to facilitate arithmetic, logical or manipulative operations, such as the address of the next instruction, a counter of the number of operations already performed, a check to ascertain the existence or non-existence of a certain condition, or a quantity or constant to be used in making certain calculations. Data movement operations from one register to another may be much faster than from memory back to memory or from memory to register. In other words, there may be a direct transfer of information from one memory location to another.

Computer peripherals

Earlier in this chapter we alluded to "input" and "output" devices. These are among the computer "peripherals" or related operating units which comprise the computer "system." In typesetting applications, one or more paper tape readers may be connected to the main frame. This enables "input" in the form of perforated paper tape to be created "off-line" and then mounted on the handiest reader to be inserted into the system. Or, keyboards could be connected directly to the computer system so that it is not necessary to create any paper tape at all. In such a case these typewriters will create electronic signals which, instead of activating a paper tape punch, will be transmitted to the computer, usually through a *multiplexor* which will take input from a number of sources and identify it with its source or origin, feeding it, in turn, to the main frame at speeds more consistent with the processing capabilities of the computer, and identified as to source so that the various text streams can be assembled into separate, identifiable "takes."

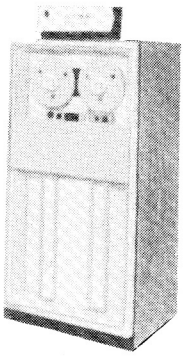
Video (or visual) display terminals (VDT's—or VDU's, for video display units as they are sometimes called) may also serve as input devices and they, like typewriter keyboards, may transmit their data either in a "parallel" fashion—that is, all the bits and non-bits in the character at once—or in a bit serial fashion—one bit at a time. OCR machines (optical character recognition devices) may be connected on-line to the computer in order to provide input from scanned manuscript pages.



A video display terminal

Some computer systems can accommodate magnetic cassette readers, which can take the output of "word-processing" equipment as input into the computer system.

Magnetic tape stations can serve both for input and for output, as well as for interim storage. Input can come in the form of magnetic tape created by another



A magnetic tape deck.

computer system. Or the computer system can generate as output a magnetic tape which could be dismounted, placed on another magnetic tape reader, and which would then serve as "input" into another computer system or to drive a typesetting device. Magnetic tape stations can be used for temporary storage of information during a computer processing cycle. The programs themselves can be "brought in" from magnetic tape, and tables of character width values can be filed away on the tape reel, to be searched for and retrieved whenever a particular type face is required. Even program "overlays" can be stored on magnetic tape. If the program is too big to be contained in memory, sections of the program not commonly required can be stored on tape and read in when needed, to overlay other program sections not then required. Strings of data which are to be re-arranged—sorted or re-sequenced—can be written to various tape stations, and passed back and forth, through memory, from one station to another, until they have been arranged in the order required. "String substitutions" or "formats" can be stored in like manner. When the program encounters a unique combination of characters it can "go to tape," search through the tape for that character combination, and replace it with a pre-stored substitute series of codes.



A moving-head disc drive with its multi-platter disc in position.

Most typesetting computers—and, indeed, most general-purpose computers currently in use—will also have one or more "discs." These discs provide a place to store programs, to store other relevant data which must be accessed or brought into memory from time to time, and to provide additional working space—a "scratch pad" for interim steps in the processing cycle. Such discs are generally "fixed-head" or "moving head" discs. The former generally offer faster access time since there is a read/write head for each track on the disc. Moving head discs have only one read/write head for each disc platter surface, and the arm on which these heads are mounted will move from one track to the next desired in order to access the piece of information or data required. Because of the time necessary to move the read/write heads from one location to another, disc access time is not as fast, but, generally speaking, more information can be stored (for a given price) on a moving head disc drive.

Discs, as well as revolving magnetic drums, provide *random access* to a supplementary store of data, whereas information stored on a magnetic tape requires a sequential search of the tape, until the desired data have been found. Because of the randomness of disc-stored data, an index must be maintained to indicate where the desired information may be found. That index may be on the disc itself or it may be kept in memory, but it must be constantly updated, so that a record of the disc, track and sector of the track will be associated with the name or number of a particular file.

Output devices.—As we have seen, a paper tape punch may be an output device. The tape produced from such a punch (or punches) can then be mounted

onto a paper tape reader (connected, for example, to a typesetting device). Magnetic tape can also be "output" and so can the contents of a removable disc platter from a moving head disc. Output can also be transmitted "on-line" to a cable-connected typesetter—thus eliminating the need to handle paper tape—either as characters transmitted in parallel, or via a *modem*, in a bit-serial fashion, for reception at a remote location.

Summary

Data are brought in for processing by various input devices. Programs must also be fed into the computer to direct a certain sequence of logical operations. How programs work will be the topic of the next chapter. It is the CPU or main frame that performs the logical and arithmetic operations, utilizing peripheral input and output devices and external storage media as well as memory so that the various commands can be executed. Channels and controllers help keep track of traffic and try to assure the efficient use of the "compute" power of the main frame. This power is determined by the size and speed of the computer and also the variety of instructions it has been wired to execute. In later models of computers some instructions can be developed by circuits that are built by programs rather than hard wiring, or combinations of wiring and programmed instructions may add to the power of the computer to implement a wider range of commands.

Review Questions:

1. How would you define an electronic computer?
2. What were some of the preliminary efforts to develop a "mathematical machine?"
3. Illustrate how electrical circuits can perform logical functions.
4. What is the difference between "synchronous" and "asynchronous"?
5. What is a "burst" mode?
6. What would happen if a computer is "input bound"?
7. What is a "unibus"?
8. What do we mean by "alphanumeric data"?
9. What is a "bit"?
10. What is "parity"?
11. Explain the meaning of binary, octal and hexadecimal. Why are they useful concepts for computer technology?
12. What is a computer "cycle"?
13. Explain the relationship between a millisecond and a nanosecond.
14. What is an "instruction set" for a computer?
15. Mention some computer input and output devices. In what category would you classify a magnetic tape drive? A magnetic disc?
16. What is the difference between sequential access and random access?

11

Programs and Programming

PROGRAMMING is a very complex subject. All that we can hope to accomplish is to give the reader an impression of the processes and procedures involved and to make him aware of the implications of what is known as the "hardware-software interface"—that is, the way in which programs interact with the computer to provide desired solutions in specific applications.

In general, there are two ways to approach the creation of a "program." One way is to assume that a particular computer with a given configuration of equipment is available: how should it be programmed for the desired application? The other approach would be to define the problem and then to determine what combination of hardware and software would best meet the user's needs. The latter implies a "systems' overview" and it assumes that the application is big enough to justify the installation of the ideal system. For the most part, "programmers" are handed assignments to perform on a given configuration of equipment. "Systems analysts" are asked to recommend the best overall solution, in terms both of hardware and software. However, the term "systems analyst" is being used increasingly to describe those who are responsible for the design of major software undertakings, even on a given hardware system.

But for the moment, as a starter, let us consider only "programming" itself, more or less independently of any hardware implications. It is customary for a programmer to start out with a precise definition of the assignment and an identification of its component elements, which is usually accomplished by "flow charting" the steps which will have to be taken.

The reader will quickly discover that the flow chart is a most valuable tool, and if he can make himself "think like a computer" he will gain new insights into many problems and will improve his ability to analyze assignments and reduce them to manageable proportions. In many situations where the installation of a computer has been contemplated, potential users have discovered that the discipline required for computer programming—namely to bring the task down into

its relevant elements—produces results which are valuable in and of themselves and may even obviate the need for the computer system.

In order to illustrate how a flow chart may be prepared and what use it may be put to, we shall take as an example the task of typing a finished paper from an author's manuscript. We shall assume that the paper will be typed manually, but that the steps to be taken would be precisely those that a computer might be asked to follow. We will find, in fact, that the computer will be called upon to make the same decisions that an individual would make, although he generally makes them subconsciously, just as he walks without thinking: "Now I will advance my left foot; now I will shift my weight from my right foot to my left foot; now I will advance my right foot; when I advance my left foot I will swing my right hand backward to provide balance and momentum."

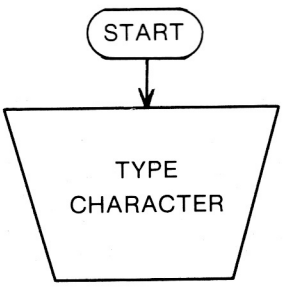
If the reader will study carefully the flow chart on the next page (and he may find errors in it, although we are not aware of any at the moment), he will see that it consists of several different routines, which may be described as follows:

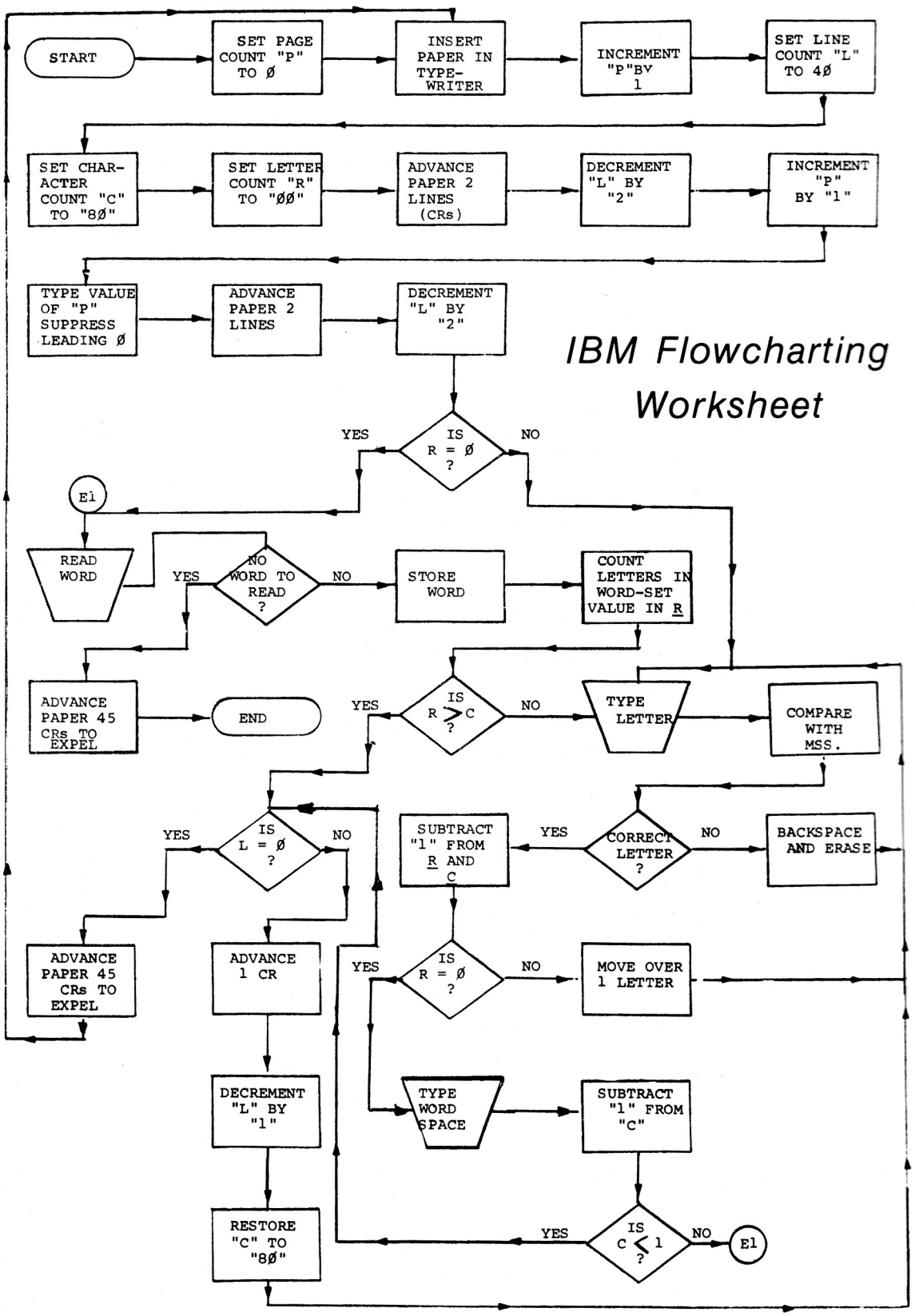
Start job and set up counters.
A page numbering routine.
A line counting routine which ejects the page when there is no longer any room on the page for additional lines.
A word reading and typing routine, which continues to read and type words until all words have been read, and which types words character by character until the next word will not fit in the line, in which case it is placed on the following line.
A letter verification routine.

These routines are interlocked. In a sense the program works as follows:

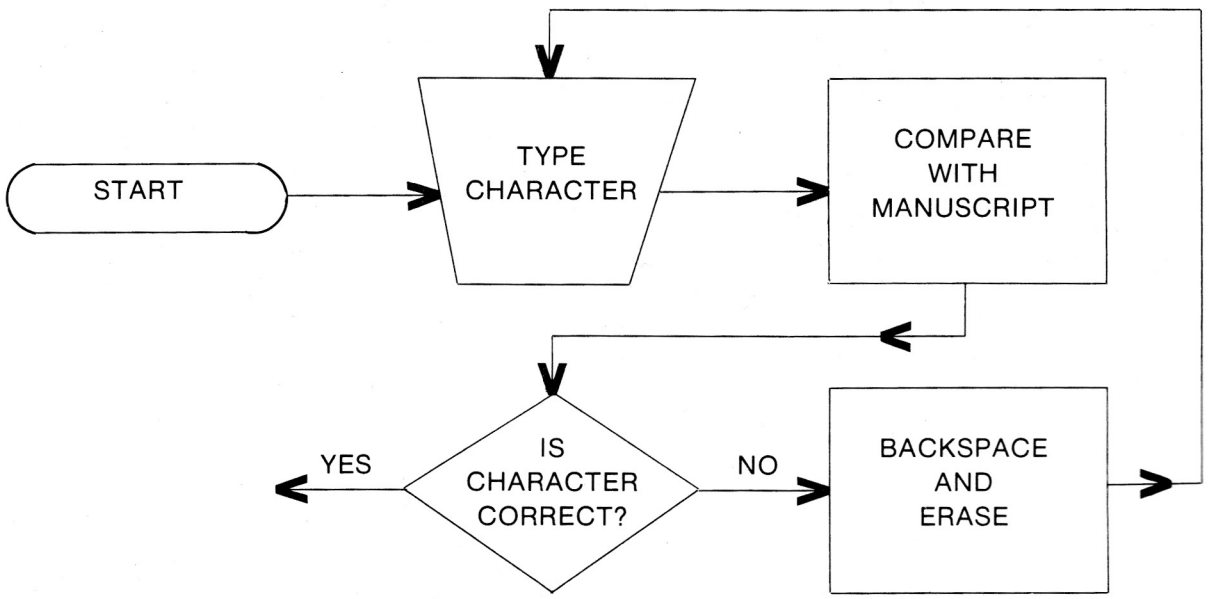
Start job: number page; go down to where the first line of copy should appear. Read a word. Determine if the word will fit in the line. Type the word, verifying each letter. Read another word. Determine (for example) that the word will not fit on the line. Determine that there is room for another line on the page. Advance the page. Advance typewriter carriage. Type the word, then read a new word. Determine now (for example) that there is not room on the line for a new word. Determine also that there is not room on the page for a new line. Eject the page. Start a new page. Number the page, etc.

The reader may be confused with the evident complexity of the flow chart shown on the following page and it may therefore be easier for him to start with something just a little simpler. Instead of starting at the beginning of our program, therefore, let us consider for the moment the typing of an individual character. The shape of the first box (a keystone shape) will suggest that this is an "input" or "output" function. In this case "type character" may be assumed to be an output function.



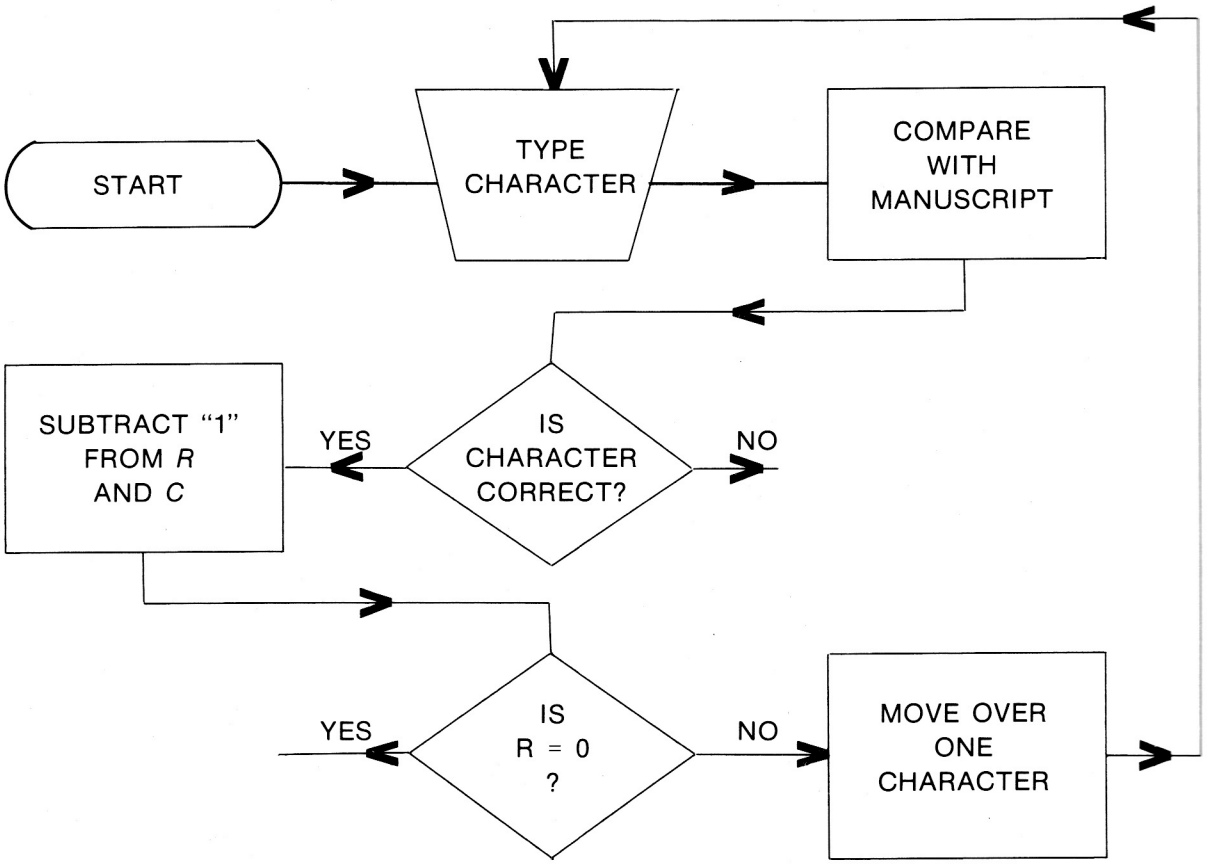

START
SET PAGE COUNT "P" TO Ø
INSERT PAPER IN TYPE-WRITER
INCREMENT "P"BY 1
SET LINE COUNT "L" TO 4Ø
SET CHAR-ACTER COUNT "C" TO "8Ø"
SET LETTER COUNT "R" TO "ØØ"
ADVANCE PAPER 2 LINES (CRs)
DECREMENT "L" BY "2"
INCREMENT "P" BY "1"
TYPE VALUE OF "P" SUPPRESS LEADING Ø
ADVANCE PAPER 2 LINES
DECREMENT "L" BY "2"
IBM Flowcharting Worksheet
IS R = Ø ?
YES
NO
E1
READ WORD
NO WORD TO READ ?
YES
NO
STORE WORD
COUNT LETTERS IN WORD-SET VALUE IN R
ADVANCE PAPER 45 CRs TO EXPEL
END
IS R ≥ C ?
YES
NO
TYPE LETTER
COMPARE WITH MSS.
IS L = Ø ?
YES
NO
SUBTRACT "1" FROM R AND C
YES
CORRECT LETTER ?
NO
BACKSPACE AND ERASE
ADVANCE PAPER 45 CRs TO EXPEL
ADVANCE 1 CR
IS R = Ø ?
YES
NO
MOVE OVER 1 LETTER
DECREMENT "L" BY "1"
TYPE WORD SPACE
SUBTRACT "1" FROM "C"
RESTORE "C" TO "8Ø"
IS C ≤ 1 ?
YES
NO
E1

Now, suppose we imagine that a person types that particular character and then looks at it, and asks himself: "Is this the correct character?" If he finds that it is not, let us assume that he backspaces, somehow rubs out the character and types in the correct one. Then he again checks to see if it is the correct character:



You will notice that the "compare" function leads to a decision. In flow charting, decisions are indicated by a diamond, whereas a rectangle indicates some kind of machine processing activity other than input/output or decision-making. You will also notice that we have diagrammed a "loop." As long as the wrong character is keyboarded and detected, the loop persists. If the correct character is struck and verified, there is a "branch" away from the "loop." In a sense, then, this "loop" could be called a "subroutine."

Suppose we have a counter ("R") to indicate the length of the word we are typing, and another counter ("C") to indicate the length of the line. Whenever the correct character has been typed, the line length counter will be reduced ("decremented") by one, and so will the word-character count for the individual word ("R"). When the word-character counter is at zero the word has been typed and a word space is needed (and that value has also to be subtracted from the line). Until the word-character counter equals zero, additional characters in the word will be typed, and this makes another loop. We branch away from this loop when the word-character counter shows a value of less than one. (It could be zero, or minus one, since we have added a space after the word, even at the end of the line.) In this case we would go to the "new line" routine.



We need not belabor our illustration any further. The reader may wish to try a flow chart of his own on some specific task, such as backing car out of the garage:

Do I have the keys to the car?
(Go to *"Get Key"* routine if answer is "no.")
Is garage door open?
(*"Open Garage Door"* routine if answer is "no.")
Open car door
Get in
Insert key
Press starter
Is car an "automatic" or a "manual"?
(Branch, alternatively, to one or the other. Assume automatic here.)
Put gear in reverse
Depress accelerator

Other subroutines may be relevant. For example, should there not be a routine to inquire if the gear is in neutral before pressing starter? What about releasing the hand brake? Some people would insist that the first steps after entering the car would be to fasten seat belt, adjust rear-view mirror, *etc.* But did the

reader notice one very significant omission? Our flow chart does not include a very necessary instruction: close door of car!

Flow charting may be carried on at a very detailed level or on a very general level. Once a particular routine has been thought through, thereafter it may be sufficient merely to describe that routine within one rectangle, even though it contains many branches and several loops of its own.

One very important aspect of flow charting is to check continually to be certain of appropriate conditions. For example, it is easier to include an instruction to see if the line counter will permit the next word to be typed even if we know that we have just gone to a new line and the word would most surely fit. This enables us to use the same loop or subroutine for all words in the line, and it will not require us to devise a separate routine for the first or first several such words. And it will also bring to light the rare case where we define the line as being exceedingly short and we encounter a word that is exceedingly long. What would we do, then, if we find that the "word" contains more characters than the line length we have requested? This could happen either because we have mistakenly specified the line length as too short (perhaps we have misplaced a decimal point in our "parameters") or some abnormality has been associated with the "word." (Perhaps we have inadvertently dropped a word space, or we have incorporated a lengthy mathematical equation into the text.) Or, perhaps we have failed to recognize that the boundaries of a word (by definition) should include an em dash to take care of the following condition: "I agree—however"—where (lacking a hyphenation routine and not recognizing an em dash as a "break point") we may have to split the linking word arbitrarily.

We can see, then, that our flow chart must consider every possibility, even the absurd or unlikely ones. We come upon the general principle of programming, that we must anticipate every contingency. We must learn to visualize the conditions we are likely to encounter, as generalities, and we must ask ourselves to consider every possible condition we might encounter, even for applications that we do not presently have in mind, but for which some day we might wish to use the same routine.

From the flow chart to the coding

Once we have developed satisfactory flow charts we need to reduce our concept or plan of action to a specific computer program. This may be done merely by "coding" if our charts are specific enough. But if we have left open many questions and choices, then an experienced programmer or systems analyst—much more qualified than a "mere" coder—will of course be required. How does the program get written?

Ultimately our program must be reduced to machine language coding: commands that will instruct the computer in the specific performance of its tasks. If we wish, we can write our original program in machine language, although this is

not done very frequently. When you write in machine language you have to keep track of every single location of every element you are working with. If you "read in" a data element and decide to store it temporarily in location 1000 then you must remember that that is where it is, and you cannot use this location for any other purpose. You must also plan your program so that all of your instructions are sequential. For example, if all instructions take one "word," your first instruction would begin in location 1001. If you wanted to add a subroutine between the first and second instructions you would have to re-number and re-assign all of the subsequent instructions—of which there might be very many.

Consequently, you are not likely to write programs in machine language. The closest you would come to machine language would be "assembly" language. Here you will use many of the same instructions and codes, but they will be somewhat less specific—especially since you will not designate memory locations but give a "name" to the area in question. An "assembler" program will later be run against your "subject" program, and it will assign specific memory locations to your instructions and data areas. When you modify your program these locations and areas will be reassigned as needed in the next "assembly." The computer manufacturer will supply you with an "assembler" which will take your "subject" program and create an "object" program in machine language. Each time you reassemble it you will get a new object program.

Your assembly language program will not be "portable." It will be written in a language which can be understood only by the "assembler" designed for the particular make of computer. If you want a more "portable" program you will want to write it in a "compiler" language, such as FORTRAN, COBOL, ALGOL, or perhaps BASIC or PL-1.

The computer manufacturer will presumably have provided you with a "compiler" which will translate the programs you write in FORTRAN, for example, into machine language for your computer. Your subject program will be "compiled" into an object program in machine language, and re-compiled as necessary whenever you revise it.

When you use these higher level languages for your programs, then you pick up many existing subroutines for certain basic functions. Programming is usually much simpler, quicker and trouble-free. You do not need to "de-bug" routines which have already been perfected by others. On the other hand, the routines others have written may not do precisely what it is that you want to do, and you may have to achieve the desired results by more roundabout procedures. Specifically, for typesetting applications, compiler programs are usually not efficient, if they will work at all. Such programs are usually intended for the solution of problems in business, economics, mathematics or science, rather than for the purpose of character-by-character processing. such as is usually required for typesetting applications. Thus the kind of programming generally taught in colleges and universities, and which is desired to help a business man or scientist gain access to a computer to perform occasional tasks, just as he might sometimes need the services of an adding machine or calculator, may not be very helpful where

"production" programs are needed which must be quick, efficient and reliable for the processing of varied input.

However, in the past several years some composition programs have been developed in compiler languages—or mostly in compiler languages, with some "patches" in assembly—which seem to be quite efficient. Better compilers are presently being written, and new computers offer faster cycle times so that if the object program happens to be somewhat slower this may prove not to be too consequential. Nevertheless, even today, the overwhelming proportion of composition programs are written in assembly.

Loading the program

When a program has been written and "debugged," so that it is available for use in a production environment, it will generally be stored in its machine language format, and it will not be necessary to re-compile it or reassemble it each time it is to be used. The stored program may be kept on magnetic tape or on punched cards, or it may remain on disc. When it is to be run, that program, or a relevant portion of it, is called into memory, sometimes by operator intervention and sometimes by a terminal command from the user. The program sets up its own registers and establishes whatever conditions are needed, and checks to see if the desired pieces of peripheral equipment are ready and available. It then accesses the first instruction, executes it, and moves automatically either to the next sequential instruction, or, if there is a branch or conditional transfer, the program contains the address of the location of the next sequence of instructions. Hence the program is "in charge," and the computer's function (if everything turns out as it is supposed to) is merely to service the requirements of the computer—such as to put on or take off magnetic tapes. If the program "hangs," warning lights usually appear at the computer console or the operator is otherwise notified. This is not invariably the case. Sometimes a program can get into a "loop," from which it can never extricate itself, and the operator must be aware of probable running times and other expected conditions so that he can diagnose problems if they seem to occur. If something goes wrong he can abort the run or try to recover, and usually will be instructed to take a "memory dump" and readings of registers so that the programmer can try to find the source of the difficulty at his convenience.

Operator diagnosis or intervention is not likely to be very helpful where "multi-programming" or "multi-processing" takes place. This occurs when the same computer is able to run several different jobs more or less simultaneously. Multi-programming describes a condition in which each user's program works within a "partition" of memory, and time-sharing describes a condition under which various users share the same sections of memory in successive, very brief slices of time. Under either of these conditions, or combinations of both, all programs run under an "operating system" and the operator's attention is directly primarily to the requirements of the system rather than the peculiarities or shortcomings of individual programs. It must generally be assumed, therefore, that all successful computer programs must be trouble-free, and they must anticipate all

conditions to which they might be exposed. They must also be designed to provide recovery points so that a job which aborts need not be started again at the very beginning. Diagnostic features can and should be built into the programs as well as ways of keeping track of the status of the job.

Documentation

Flow charts provide the basic clue to an understanding of the program, but a more detailed explanation is essential, and such documentation can be provided by the programmer as he does his coding. Explanatory comments may accompany his array of instructions, and these, in themselves, if produced conscientiously, may be sufficient to enable another programmer to understand what the first has tried to do. As an illustration of IBM 370 assembler coding, here is an example of a printout which shows the "object" or machine coding (the first five columns) generated from the source coding by the IBM assembly program, from assembler coding. "STMT" means *statement number*. The actual assembler coding consists of the data to the right of the statement number. "S4LOOP" is a tag assigned to a subroutine.

```
  LOC   OBJECT CODE       ADDR1  ADDR2  STMT        SOURCE STATEMENT
000790  4170  B084              020A4  602           LA    R7,TABLST    POINT TO FIRST ENTRY
000794  5860  C070              01090  603           L     R6,NUMTAB    GET TABLE COUNT
000798  5850  C074              01094  604           L     R5,TABADD    AND ADDRESS OF AREA
00079C  9500  7002       00002       605  S4LOOP   CLI   2(R7),0       R1 OR R2 FAMILY?
0007A0  4770  D7B6              007D6  606           BNE   S4SKP        SKIP IF NOT
0007A4  D200  505D 5000  0005D  00000  607           MVC   X'5D'(1,R5,0(R5) MAKE EN FIGURE WIDTH
0007AA  4155  0080              00080  608           LA    R5,128(R5)   TWO TABLES IN R1
0007AE  D200  505D 5000  0005D  00000  609           MVC   X'5D'(1,R5),0(R5)  DO IT FOR R2 ALSO
```

LA means "load address"
L means "load"
CLI means "compare logical immediate"
BNE means "branch not equal"
MVE means "move character"

The tags which follow these instructions designate registers or routines. The final column contains explanatory comments by way of documentation.

This particular piece of programming was taken from a routine used to calculate width tables.

Review Questions:

1. What is a "systems analyst"? A "programmer"? A "coder"?
2. What is a flow chart? How detailed must it be?
3. Select some pattern of activity and see if you can draw a flow chart of all of the steps involved. If you are a linotype machine operator, for example, you might try to describe your activities in such a chart, beginning with machine set-up.
4. What is "machine language"? "Assembler coding"?
5. Why is a "compiler language" more "portable"?

12

Some Introductory Thoughts About Typesetting "Systems"

TYPESETTING DEVICES have to be understood and evaluated as part of a "system," with printing as the end product. The need for additional end products could of course affect the validity of the system and obviously the viability of a system depends upon the manner in which the components fit together. For example, if the printed product must be produced by letterpress, photocomposition has, in the past, offered a less logical typesetting solution than hot metal, since in the latter case printing can be done from the type itself (in short-run work) or from plates made from molds taken from the type (in long-run work).

The system must also take into account the work flow—the total organization of production from the time the manuscript is received until the printed output has been delivered. The intensive time span for critical production must also be considered. One typesetting system may be good in one application but not in another, depending, for example, upon whether the customer needs to inspect the product at intervals as it is being created, and whether or not he will revise it substantially while it is in an interim stage.

The system also has to take into account the typographical ingredients themselves—the variety of type faces and sizes, the format requirements, the amount of illustrative matter, etc. A method which is entirely appropriate and necessary for the production of scholarly treatises in advanced mathematics may be completely inappropriate for setting newspapers or business forms.

Moreover, there is increasing emphasis on the development of systems which contemplate including the author or writer (as well as the editor) in the "loop" and where this is possible the "system" clearly goes beyond typesetting and becomes at least an "editorial" system, if not—as with some newspapers—an "electronic newsroom."

In view of the complexity of the problem, how can we discuss typesetting and composition systems in an intelligent and reasonably comprehensive manner?

Obviously we shall have to approach the topic by a consideration of prototypes or models. Our generalizations will be relevant only to the set of assumptions that the model implies.

The hot metal shop—linecasting for book work

Let us assume, then, that we look first at a typical linecasting operation, perhaps engaged in producing books—a relatively specialized application. Here are some of the critical steps in the process.

1. Manuscript is received. Specifications are studied. Copy is marked up, divided according to type sizes and measures required, and assigned to various operators, through their foreman.
2. Material is keyboarded. Slugs are assembled into galleys. All miscellaneous takes of different sizes and measures of type are arranged in sequence by hand compositor or "bank man." Footnotes are positioned at the end of relevant paragraphs. Proofs are then pulled.
3. The manuscript is re-assembled and the proofreaders check galley proofs against manuscript for the "first read." "Typos" (typographical errors, printers' errors or "p.e.'s") are marked according to proofreading conventions. Other conditions of the manuscript or galleys may be queried.
4. Marked galleys are returned to the foreman of the machine department so that corrections may be assigned, either to the original keyboarder or to others. These correction lines must be keyboarded in the appropriate type faces, sizes, and measures.
5. Proofs are usually pulled of just the new slugs. The corrected lines are then inserted into position by the bank men and the new galley proofs are pulled.
6. The new proofs, the old proofs, and the proofs of the corrected lines only, are returned to the proofroom for checking, together with the manuscript. If some errors still remain the cycle will be repeated. A few errors may be permitted to go uncorrected, though marked, awaiting further revisions by the author. Queries raised in the initial reading must be transferred to the new galleys.
7. The manuscript and the new marked-up galleys go back to the publisher, who usually forwards them, unread, to the author.
8. The author reads them carefully; he catches a few undetected typos (the end product will never be entirely error-free) and responds to the proofreader's queries. He changes his mind when he sees his material "in print," adding revisions to the text in the interest of clarity, felicitous expression, or to incorporate later information. These are called "a.a.'s" for "author's alterations." He makes these revisions on the galleys or on the sheets appended to them.
9. The material goes back to the publisher, who returns it to the typesetter. The marked-up galleys and other supporting documents are sent to the composing room and the necessary lines are reset.
10. These are proofed, then inserted into the galleys, and all proofs go again to the proofroom. A clerk, secretary, or expeditor may keep track of the

number and nature of a.a.'s and the amount of time or the extent of resetting necessary to incorporate them.

11. After the revised galleys are checked again, they are returned to the publisher and usually to the author so that he can see that the corrections and revisions have been made.
12. If there are no further changes, the compositor then proceeds to "make up" pages. He knows the number of the first folio, and the depth of the desired page. Running heads must now be keyboarded, as well as the folios. The makeup man breaks the galleys into pages, moves footnotes to the bottom of the page, inserts the running heads and folios, spaces out around heads to make pages balance, and moves lines from page to page as necessary to avoid "widows" or illogical page breaks.
13. After pages are made up, page proofs are pulled. These go back to the proofroom to check for balance of pages, spacing, correct running heads, proper front matter, page numbers on the table of contents, etc.
14. Assuming everything is correct, page proofs are sent to the publisher.
15. Usually the publisher's copy editor or proofreader now reads the full text for the first time since it has been typeset. He may revise the text himself to make the book fit the desired number of signatures, or he may ask the author to do so. The author gets another look at his product, but he is discouraged from making further revisions (except perhaps at his own expense!).
16. Revisions nevertheless occur in the made-up pages. They have to do with the aesthetic appearance of the page, the manner in which illustrations are positioned, where heads fall, and how the book divides into signatures. Such revisions can be substantial or minor. Marked-up pages go back to the composing room, through machine composition, makeup, proofreading, back to composition, and finally pages are completed and supplied to the printer or negatives are provided for platemaking.

The system from the point of view of a magazine publisher

In order to visualize procedures from the standpoint of a magazine publisher, it will be helpful to reproduce (with the permission of the author, Jerry A. Carlson, then managing editor of *The Farm Journal*) a portion of his analysis of current production techniques, taken from his unpublished technical report (1970), entitled *Magazine Writing and Computer Video Terminals*.

> Tracing a typical article in *Farm Journal* will give a fair indication of the amount of editing that goes into a story before it reaches print. Glenn Lorang, one of *Farm Journal's* field editors, phones in from Spokane, Washington, suggesting a one-page feature on farmers who are starting their own grain-hauling fleet of trucks to bypass the railroad's high shipping rates. The managing editor assigns the story, and for several days the reporter fills notebooks with quotes and facts. He roughs out a manuscript on his office typewriter. The preliminary draft is composed of copy paper taped or glued together, paragraphs circled and marked "insert at A below," heavy pencil tracks throughout. Without going to the trouble of an actual word count, he can only guess as to its total length. Next morning his secretary retypes the article. Glenn checks it for typographical errors and mails the corrected manuscript to *Farm Journal* headquarters in Philadelphia. It arrives in the managing editor's office three days later. The final character layout specifies room for 80 lines averaging 52 characters. That is about 70

lines shorter than the original manuscript, so it will need severe cutting to fit. It also needs other editing, so several sections must be rewritten. The rewritten version then goes to a secretary who types a clean copy. This copy, done in five carbons, receives further minor pencil editing and typographical markup. Finally, it is ready to be sent to the printer. Fortunately, the airline pouch service out of Philadelphia is not delayed and the story arrives in the Chicago printing plant the next morning. The foreman of the plant's composing room carries the copy to the Linotype section about 120 feet away. The article then waits its turn at the Linotype work center for about three hours, then the Linotype foreman hands it to a Linotype operator. The operator keyboards the text in galley form.

The lead galley is carried to a proof press, several copies run off, and then the type itself is assigned a rack location to be stored while the galley proof awaits its turn in the proofreading room.

A proofreader matches the galley with original copy and marks typesetting errors on the galley proof. New corrections are set for the galley, which is then sent to the composing room section. The composing foreman parcels out the correction work to a specialist. This "comp" first inserts the galley corrections in proper position, then begins fitting the type into page form to match the artist's specifications. It takes about an hour to compose the page. With a final hammering down of type and locking of quoins, he shouts for a proof boy to run several white proofs at the nearby hand press. The proofboy is somewhere else, so the comp, with appropriate descriptive phrases, runs off the page proofs by himself.

Later, the comp foreman hand-carries this proof and others 200 feet to the proofreading room. The proofreader marks all the errors she finds on the proof, then copies by hand all the changes again on a duplicate proof, since one copy will be going out to the customer and another retained at the plant as back-up.

Now signed proofs and increasingly smudged, dog-eared copy is sent back to the composing room foreman, who removes one signed proof, puts it in the air express pouch for delivery to the authors, and piles the remaining material on a shelf. The pouch containing the proof will be 800 miles away in Philadelphia next morning, if the messenger service works well between airports on both ends, if the planes are on schedule, and if the airline people do not bury the pouch in an unknown storage igloo under tons of other air freight.

The editor, busy getting three other stories into shape for sending to the printer, sees correcting the proof as an unpleasant chore which can be delayed until the last minute. That time arrives quickly, and the editor's pencil comes down on the carefully typeset and corrected proof. Six lines too long! But he deletes a couple of words in each of three paragraphs, pulling up three hangers and shortening the article by three lines. One sentence that is not too essential is in the top of a long paragraph. Cutting it would mean resetting the entire paragraph. A more essential sentence near the end of a paragraph could be dropped without very much expensive resetting. He elects for editorial integrity and deletes the expensive-to-cut sentence. That is two more lines out. Here is another possible hanger to cut: If he deletes these two words, will that two-word hanger come up to the line above? He measures with hash marks on a slip of paper as a ruler. Maybe it will fit, if the lino sets it tight.

Again, the pouch courier service willing, the proofs travel between Philadelphia and Chicago. The composing-room foreman checks over the editor's proofreading corrections, watching for any proofreader's queries the editor may have missed. He brackets in red ink the lines that will have to be reset—in this case about 10 percent of the story—and carries the proof to the lino section. Corrected slugs again await a composing specialist, who inserts them and cleans up the page for proofing. The proofreader again goes over this hopefully final proof—now on yellow paper instead of white—and if all is well, the page will be ready to send to the foundry for press plate making.

By now, the final yellow "foundry" proof has arrived in Philadelphia. Somehow, this yellow color stimulates an editorial scrutiny that the white proof could not. It is a last chance to fix an oversight before the presses roll. In this case there were no errors or changes of heart after the plate was made. Many articles are not so clean and fortunate, which means a second plate must be made—a "Number Two" that usually costs the publisher roughly twice as much because of foundry overtime charges.

Perhaps one proof out of five also goes through a "revise" stage, where the corrections are so heavy an editor cannot count lines and achieve a fitted page the first time. In a big, efficient job shop where every customer waits his turn, that can add a day of time before foundry plates are made. A substantial share of an editor's time—perhaps 30 to 50 percent—will be spent with such mechanical copy fitting and related processing chores.

The author-publisher typesetting relationship

It may be seen, then, that a "system" approach to typesetting cannot exclude from consideration the relationship between the typesetter and the publisher, or the relationship between the publisher and the author.

"Under the present technology," wrote Leonard Shatzkin in *Publishers' Weekly*,[1] "—and ever since Gutenberg—the only way to be sure of what was happening in composition was to check the results at each step. The naive could talk about the desirability of a perfect manuscript. But the initiated knew that even with a perfect manuscript the compositor could easily strike the wrong key. If he struck the right key, that channel of the Linotype machine could contain the wrong matrix, or if it had the right matrix, it could be raced out of position by another matrix as it dropped toward the assembling elevator. One operator might set loose lines with excessive spaces between words, and the man next to him, who did half the book or set the corrections, might set too tightly.

"Then there were—I should say there are—the hazards of type metal too hot, so that it is too weak to print well or mold well, or too cold, so that the fine serifs are lost and the surface is not smooth. Perfectly good slugs can be dropped, or transposed or lost. If more than one man is working on page makeup, widows in the front of the book may be dealt with differently from widows in the back of the book. In short—if you care about quality—there is really no way to escape the need to verify at every step of the process. Hence the practice of supplying the customer with galley proof, page proof, stone proof, repro proof, and blueprints.

"With what result? How often have you seen a book completely free of error? Well, one result is this. Proof is unavoidably necessary to catch errors in transcribing the manuscript, but it provides the opportunity for reconsidering the manuscript, in type. We all know how low is the resistance to that opportunity. That is human nature and virtually uncontrollable so long as the opportunity exists. As a matter of fact, we are so used to the idea of doing this—which we call author's alterations—that we imagine that the proofs are submitted for the purpose of author's alterations, rather than catching printer's errors, which is, of course, complete nonsense.

1. "Some Effects of Computer Composition on the Editorial Process in Book Production," *Publishers Weekly*, April 7, 1969, pp. 28 ff.

"While the printer pays for correcting his own mistakes, the publisher pays for these author's alterations. How expensive are they? Well, at McGraw-Hill they ran, for the year I totalled them, $456,000. We can certainly put that money to better use."

Shatzkin goes on to analyze the nature of author's alterations and finds that, in general, they divide themselves into two categories:

1. Changes that could easily have been anticipated in the manuscript and did not require typesetting to become apparent.
2. Changes that simply represented two points of view, with the second not better than the first. The copy-editor or the author hyphenated—the proofreader unhyphenated—or vice versa.

"So," Shatzkin continues, "the first cost of verification by seeing proof is *money*. The second cost is *time*. The cost in time is greater than you might imagine, because it extends far beyond the time required to make changes."

> Predicting the precise time that proof will be returned is impossible, and even if it were possible, the amount of work to be done varies greatly and can only be estimated by looking at the proof itself. That is why no intelligent compositor attempts to schedule composition corrections in advance. In place of scheduling, he (and the publisher, for that matter) resort to queuing. When the work comes in, it takes its place in line—and the more work in line the better the assurance of an even flow and lower cost. That is where the book spends most of its seven manufacturing months—on a series of waiting lines at the publisher, the author and the printer.
>
> Actual production time in a manufacturing schedule of seven months is probably less than one month, which may indicate how costly in time is our verification through proof.

How can this arrangement be changed? Shatzkin concludes by stating:

> It is the possibility of moving book manufacturing from the handicraft stage closer to the production line—from the whittling by designer, production manager, compositor, printer, binder, passing the willow stick back and forth among the group, each taking his turn—to an automated production line with controlled and predictable results—it is in this possibility that computerized composition offers its greatest hope.

In terms of the relationships between author, publisher and compositor, we can visualize a complete restructuring of responsibilities. We shall appraise them as we go on. We shall find that one approach implies not passing on the manuscript to typesetting until it is perfect and all the specifications have been determined, and even coding for automatic processing. A second alternative would perhaps afford even more opportunity for author and publisher intervention than now exists for book composition, by on-line interactive computer programs and systems.

But in either case there will be a change, and it is necessary not to lose sight of the implications of this fact as we proceed, because it means opening a new kind of dialogue with all of those who are involved in the process, and they must all become familiar with the opportunities and the choices.

A modern newspaper system

Over the past several years it has become customary to look at the newspaper industry as a pattern of things to come. While automation of the editing and composition processes was long delayed in newspapers because of the hot metal orientation of at least the largest of them, this condition has changed dramatically since about 1974. In part it is because the relationship between the editorial process and composition activity is relatively well defined and homogeneous throughout the industry—which is rarely the case with other segments of publishing. In larger part, perhaps, it is because writers are "captive" in the sense that they generally work in the same building so that their keystrokes can easily be captured and they can also be subjected to some small degree of procedural discipline. But for whatever reason, including economic pressures, and the size of the market to potential vendors, a "system" concept has gradually emerged within the newspaper industry—a concept which embraces the entire editorial and production process. Ultimately this concept may even be extended to include, as well, many related business functions, as well as those having to do with the storage and retrieval of information from the newspaper's "library" or "morgue."

Today's newspaper is thoroughly committed to the use of the video display terminal (or VDT) for initial input of reporters' copy. The reporter sits at his terminal much as he would at a typewriter, cradling the telephone to his ear, and takes notes of interviews, building up a scratch pad of information which he can reference at his convenience. This is filed under his name and can be accessed on the screen in a matter of a split second simply by typing the appropriate command. Then the reporter begins to construct his story. He may revise it as he types, and he can use the terminal to manipulate text by moving paragraphs, searching for key words, or making "global changes" in terminology if this seems indicated. When he is satisfied with the story he can "route" it to his supervising editor. This is done "by the system"—usually by his merely pressing the "send" key. The name of the story will then appear upon the editor's directory, with an indication of the approximate story length and the name of the writer. By positioning his "cursor" at that location and pressing the "get" or "fetch" key, the story will immediately appear on the editor's screen. He is able to "scroll" through this "file" and make whatever changes he desires, even incorporating side comments (as to his own superior), and he can then "send" the story to be hyphenated and justified to whatever measure and point size he wishes. In many systems the story will come back (again almost instantaneously) on his screen so that he can review its precise size and appearance. He can then "send" off his story to someone else for further review, editing or comment, or he can command that it be typeset. In the latter case, the "file" will be queued for the on-line typesetter and will very shortly appear as a galley to be stripped into position on the indicated page. As of this writing, systems are also beginning to emerge which would route the story next to a page make-up man who would also work with a video tube on which a diagrammatic representation of a particular newspaper page would be displayed. This page editor can indicate desired locations of stories, and when he is satisfied, command that all stories on the page be composed at the

same time, to avoid any hand make-up or stripping. High-speed cathode ray tube typesetters, capable of setting full-width newspaper pages, presently exist and can be commanded to call out the desired type faces—of whatever size or variety—without any manual intervention.

While all this is going on, other editors are scanning through wire service listings, already stored on disc as they flow in from the various agencies, deciding which stories to use, calling them up and editing them, or even combining two stories (from different sources) by making side-by-side comparisons on their "split" screen. These stories, when so selected and composed, would in turn be routed to page make-up or directly to composition.

And in another area of the newspaper, girls sit in front of VDT's entering classified ads as they are dictated over the telephone. Their screens capture these ads, perform credit checks against the persons placing the ads, and direct the ads to be sorted or sequenced for those days on which they are to appear (taking into consideration "skip dates") until a command is given to "dump" the day's classified ads. They will be typeset all at once, at very high speeds, and in the proper sequence. Billing information will also be generated by the program.

What a contrast between the kind of relationship described by Carlson or Shatzkin and that within the modern newspaper! Of course, the experiences described for magazine and book producers pertained to hot metal, and to non-computerized composition. But today, because of the availability of new tools, everyone is taking a fresh look at the possibilities, and these must, of course, be interpreted in the light of the specific problems, provided the definition of such "problems" is sufficiently broad and sufficiently inclusive. As we shall proceed to a consideration of composition software and systems we shall have these problems very much in mind.

Review Questions:

1. For the kind of typesetting you are most familiar with, can you make up a flow chart of the "system" in present use? Include the author and publisher (editor) in the process.
2. Do you really know what goes on at the publisher's before he gets the manuscript? Could you offer to the publisher alternative ways of helping him to perform some of his tasks? Do you think it is relevant to re-study the relationship between the commercial typesetter (or printer) and the publisher? Do you think there would be a sympathetic response to this suggestion?
3. Where would you look for change to come from? From the publisher, demanding it? From the composition supplier, suggesting it? From the manufacturer of equipment, laying out guidelines for new processes? If not from these sources, what other internal or external stimuli for change can you visualize?

13

The Composition Process

IN SUCCEEDING CHAPTERS we will be dealing with the functions of "composition," and the "composition cycle," so far as computer programs are concerned. We shall not be immediately concerned about that portion of the composition process which constitutes "input," nor shall we be considering the problems posed by final "output" to the typesetter. At this point our only interest in the typesetter *per se* would have to do with its "front-end intelligence" when it is used as computer to perform various software tasks. For convenience in our exposition, then, we shall assume that these various software functions will, in fact, take place within a "stand-alone" or "free-standing" computer—that is, one which is not buried in a keyboard, or an editing terminal, or in a typesetter, or in any other device (such as an optical character reader, or an ad markup or page layout terminal). In point of fact, of course, many or most of these composition functions could indeed be performed in one place while certain others are performed elsewhere. But, in general, "stand-alone" computers or computer systems are more powerful and can perform a wider variety of composition functions than is the case with mini- or micro-computers which are also used as terminal- or typesetting-controllers. But this is not invariably the case.

At one time all so-called computerized composition functions had to be performed within a "stand-alone" computer, for there was no "intelligence" built into these other components—keyboards or typesetters—and there were no editing terminals or OCR devices which might be endowed with the capacity to perform such functions. Of course, this is no longer the case, but the functions themselves remain pretty much unchanged. The main categories of "composition" software have to do with:

- ***Justification***—which is a broad term for arranging text horizontally across the line. It is concerned not only with running text, but also with the composition of "display."
- ***Hyphenation***—which is a sub-set of justification concerned exclusively with the question of how words should be divided ("split"), when necessary, in order to achieve better justification.

- ***Pagination***—or other methods of composing text vertically as well as horizontally, so that text is properly spaced out to fill a page, and appropriately divided into pages.
- ***Editing and corrections***—which are directed to the elimination of input errors, or to resolving problems created during the composition or pagination processes, or to improving the final product by introducing changes and "author's alterations" after the initial text has been "input" into the "system."
- ***Operating system and File Management system***—These vital supporting programs become increasingly important as video terminal-oriented systems become more common and more useful.

Because *editing and corrections* may take place at various stages in the composition process, much of what we shall have to say on these particular subjects will be scattered through the remainder of this book.

Our first task, then, is to consider "justification." We may get some insight into the process by examining the manner in which this book is being typeset, although we are not, in fact, using a "stand-alone" computer for this purpose. Nevertheless, there are some similarities.

In typesetting this book we are using VariTyper's Comp/Set 4510, which is known as a "direct-entry" or "direct-input" typesetter. This means that we are keying the input from our manuscript directly into the line buffer associated with the typesetter, and there certain calculations are performed in order to justify the line—that is, to produce sequential lines which "run on" as "running text" where "line-ending" decisions must somehow be made in order to determine how much material will fit within the line. Then the typesetter itself will do the "justifying" or spacing out, to provide even left- and right-hand margins.

The operator mounts the desired font disks, and hits the "reset" button which causes the device to read and store the set width values for the individual characters as they are represented on the film disks by bar codes. In some models, not all of the width values are stored—in the interest of saving "memory"—but most of them are. The others are read "on the fly" when they are required.

The operator must then fill in the blanks at the top of the screen. He indicates the desired point size, the font (fonts 1 through 4) for each of four disks—(*i.e.*, A1 through D4), the line length, in points and picas, the "primary" leading to be associated with each line, the "secondary" leading which may be used from time to time. He also indicates whether he wishes the machine to be operated in an "automatic" or a "manual" mode—that is, whether he wishes the machine to make "its own" end-of-line decisions if it can do so, or whether the operator wishes to make them.

The operator also indicates the minimum acceptable interword space value. We are on an 18 unit-to-the-em system, and we have set this at "4." The maximum

interword space value we have set at "12." And we have assigned a letterspacing value of "zero" since we generally prefer not to letterspace.

And so, to produce this book, we are now setting in font C1, size 11, on 13 leading, with from 4 to 12 as the interword space expansion potential. After keying in the line length, the machine computes that there are 629 units in the line, and displays this under "LR"—line remainder—at the top of the screen. It derives this calculation in the following manner. In 32 picas there are 384 points. Dividing the point size by 11 yields 34.9 eleven-point ems. Multiplying 34.9 by 18 units per em should produce about 628.2 units, but it rounds this off to 629 units.

As we key across the line in the manual mode (wherein we have to make our own end-of-line decisions) the counter will display the decrementing of character widths. When we reach the first point at which the line is within justification range based upon the largest definition of the interword space—in our case, 12 units—a "beep" will sound and a blinking symbol will appear at that location. But we can continue to key, watching the countdown, which shows us how many relative units remain in the line. When we see that we cannot fit another word into the line we can strike the "return" key. The line will be terminated and the machine will proceed to justify the line as it typesets it (flashing characters onto the photographic paper) at the same time that we can continue to key the next line.

Here is a line which has been justified with the maximum interword sp
Here is a line which has been justified with the maximum interword spaced exceeded

The first of the two lines immediately above shows how much fits within the line on the basis of the 12-unit count. The second line has brought the interword space value down to the point where not even a period can be added to the line.

When the computer processes.—Substantially the same thing happens when a stand-alone computer performs the processing of input which does not contain end-of-line decisions—so-called "idiot tape." But in most such programs the count down imputes the minimum interword space to the spaceband value and continues across the line until it has been overset. Then it backs off to the end of the preceding word. If that word falls within the justification range, the program then terminates the line.

To achieve this, the computer is instructed to accept what amounts to a continuous stream of data, although if the input were magnetic tape it would be "blocked" so that there is a tape gap after a certain number of codes have been recorded. The codes which will be read and processed, in either event, are basically of two kinds: instructions to the computer, and text material. The first category consists primarily of "tags" which identify typographic "situations" or conditions, such as paragraph endings, or the beginning and ending of display heads. It is the function of the program to be able to distinguish between such flags, tags or signals (which are intended to activate certain processing routines) and the balance of the input (which is the text matter that the instructions relate to).

Usually the first block of information to be processed has to do with setting up the job. Perhaps there will be a job title, or an operator number or some other type of identification. This is useful for record-keeping purposes, but it may also be a key element in determining whether a later run is being processed against the proper "file." (We shall use the word "file" from time to time to describe the store of text data and related codes either in process or available to be processed.)

Immediately after the identifying material, the program will look for the line length, point size, and font and leading information. This may be expressed as one "combined command" in which this information is provided sequentially—as for example: [cc32p,01,00,12] or as separate commands: [LL32p] for a line length of 32 picas: [TF01] for type face #1; [PS11) for point size, and [BL13] for body leading of 13 points. Notice that the program may require a "begin command" and an "end command" flag for certain kinds of arguments, but not for others. This will vary according to the conventions of a particular program. For example, @rr could mean enter "ragged right" and @er could mean "end ragged."

As with the Comp/Set, the first task of the computer software is to reduce the line length to units of measurement which will correspond with the set width values for the type characters we are going to try to fit into the line. We can elect to express and compute type widths in terms of an absolute unit system (which is converted into millimeters or thousandths of an inch), or even, perhaps, in tenths of a point. (Thus a six-point em would consist of 60 units, and a twelve-point em would consist of 120 units.) We might, if we preferred, adopt a relative unit system so that an em of whatever size would be made up of 18, 36, or 50, or 60 or 100 units. In either case (relative or absolute unit values), there are special problems which will be encountered in the conversion of character widths to units, especially where different type *sizes* are intermixed within the same line. But most important of all, a "good" composition program would be written to be "output device-independent"—which means that the relative unit system for *any* typesetter could be accepted by the same program, given proper "set up" instructions.

We shall revert to the implications of these counting procedures at a later point in this discussion. For the moment, however, we shall assume that we have established a line length (32 picas) and a point size (11 points), and a unit count based on 18 units to the em, using [TF01] to indicate the roman text type face. We shall imagine that the computer has now processed the parameter definition, has been instructed as to the desired interword space values or has used its own "default parameters," has now read the first identifying blocks and then this much of the text stream into a work area of memory which we shall designate as area *A*:

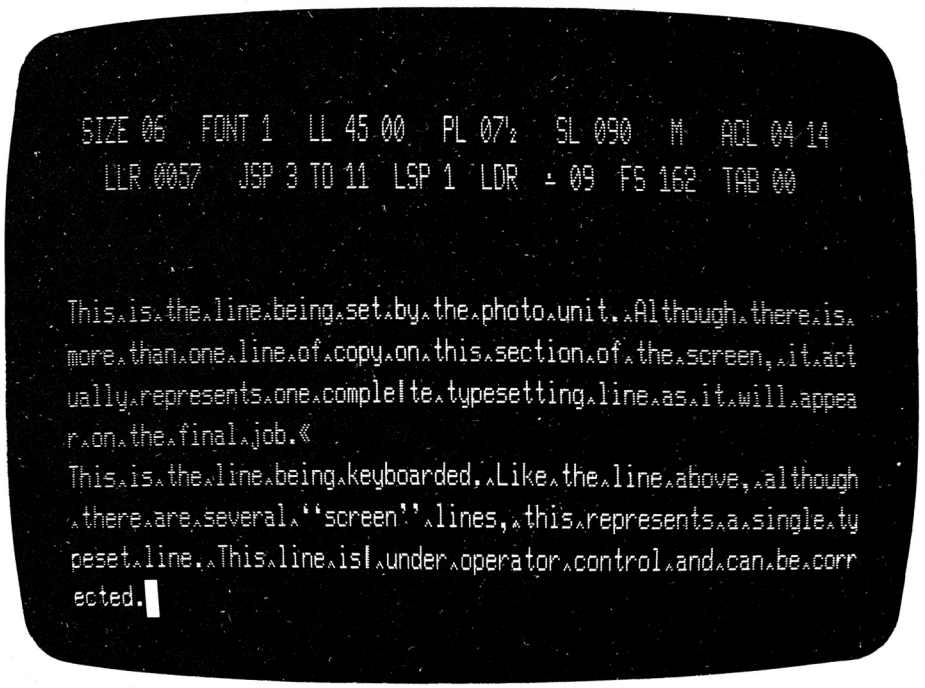
[cc32p,01,11,13][em]This is the first portion of input into our typesetting program and we are ready to begin processing.[ql]

(We have used [em] to indicate the em space at the beginning of the paragraph and [ql] to indicate a quad left at the end of the paragraph. Normally we would be able to generate these codes with one keystroke and they would be carried on the input tape and within the computer as one code each.)

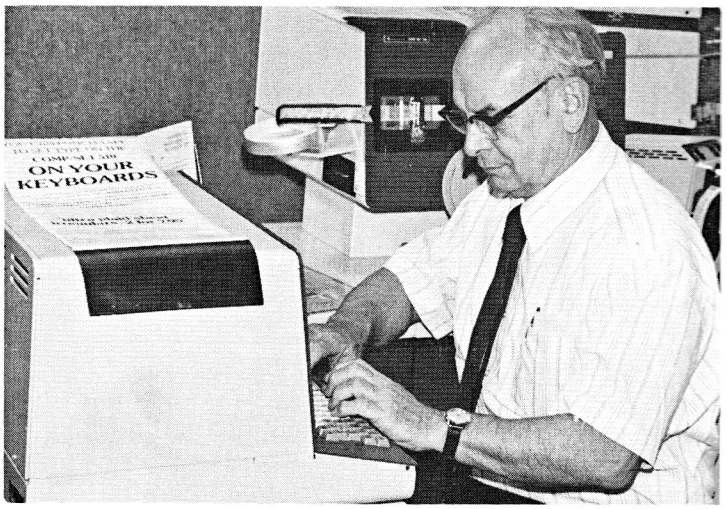
The initial calculation which the program must go through is to determine, just as with the Comp/Set, the number of 11-point ems which fit into the 32-pica line length. Let us assume that the program reaches approximately the same conclusion, but it will be easier for our illustration to assume that it comes up with 630 relative units.

It will now be necessary to bring into computer memory one or more tables of width values. For the moment let us assume that we have somehow loaded into memory one table which establishes the width values for text type face "01" in which we are going to begin to set. Using an octal ("six-bit") coding structure in order to keep our example simple, we may imagine a table which reveals that the code which represents an "em" (assume it is 010 000 or an octal 20) has a width value of 18 units. Consequently, the program writes the code for an em space into another word area, *B*, which corresponds to the output buffer, and decrements the line counter by 18. It goes on to the next character, which is a capital "T." Since we have supposed a six-bit or six-level input system, the program learns of the fact that the "T" is an uppercase character by first encountering an upshift code. The program therefore goes to the upshift portion of the width table, and learns that the capital letter "T" has a unit value of 12. It writes the "T" to the output buffer and decrements the counter by 12, so that it now stands at 600 units.

The table look-up procedure just described is more complicated than appears on the surface. Actually, the em value of 18, or the "T" value of 12, which is stored in the table or tables, may best be represented by a single character. If we are dealing with an 18 unit system, we may only need 18 characters to describe the various widths, and we could store in the "box" whatever symbols correspond



This is a representation of the Comp/Set 500 screen showing two lines. The first has already been set. Notice the small cursor within the word "complete" indicating the point at which (given 11 units) the line began to be within justification range. For the second line of text, which has yet to be typeset, the beginning point of justification follows the word "is." These lines are exceptionally long since they are to be set in six-point type across a 45-pica measure.



This is a photograph of the author keying on a Comp/Set 500, with the 503 paper tape attachment shown in the background. In point of fact, this particular machine is the 510, which sets in sizes up to 74 points. The photo was taken at a trade show.

to the octal numbers 000 000 through 010 001 to represent 1 through 18. Let us suppose that we have assigned for the first 18 codes the numbers 0 through 9, and the letters A through H. Thus the width of the em would be expressed by an "H"—which in this instance would be interpreted as "18"—and the width of the "T" would be expressed as a "B", which would mean 12.

The example which will be found on the next page illustrates the process at work. It will be noted that the program continued to loop through the cycle of moving characters into the output buffer and decrementing the counter until the counter went negative. Then it was necessary to reverse the procedure, adding characters back on, in the reverse order, up to and including the preceding spaceband. At that point, then, there is a tally of 37 relative units "left over" and there are 13 spacebands over which this surplus—it is usually called a *deficit* since it represents the amount by which the line is short—can be distributed. Dividing 37 by 13 spacebands yields something more than two units per spaceband and less than three. If the function of the program is merely to bring the line within justification range, then this line can passed on for output (when the job is completed) with the knowledge that it will justify within the specified parameters. But if the program must "put out" the specific value required for each interword space, it would call for seven relative units instead of four for the first 11 interword spaces, and six units (instead of four) for the remaining interword spaces.

But suppose that when the program "backed off" the last word because the counter overflowed, the counter remainder (divided by the number of spacebands still left in the line) produces a quotient greater than the indicated desired maximum interword space. The program would then call for the hyphenation routine in an effort to discover whether there is a portion of the "straddle" word that could be made to fit in the line. If, however, the word could not be hyphenated, the program would have to release the line as a "loose" line, leaving it to the discretion of the typesetting device (if it actually accomplished the final justification), or exercising the remaining parameters of the composition software in the computer (if it was charged with the distribution of the deficit) to decide how

Character	*Character width*	*Line count remainder*	*Number of spacebands*
		630	
Em	18	612	
T	12	600	
h	10	590	
i	5	585	
s	9	576	
s/b	4	572	1
i	5	567	
s	9	558	
s/b	4	554	2
t	6	548	
h	10	538	
e	9	529	
s/b	4	525	3
f	7	518	
i	5	513	
r	6	507	
s	9	498	
t	6	492	
s/b	4	488	4
p	10	478	
o	11	467	
r	6	461	
t	6	455	
i	5	450	
o	11	439	
n	11	428	
s/b	4	424	5
o	11	413	
f	7	406	
s/b	4	402	6
i	5	397	
n	11	386	
p	10	376	
u	10	366	
t	6	360	
s/b	4	356	7
i	5	351	
n	11	340	
t	6	334	
o	11	323	
s/b	4	319	8
o	11	308	
u	10	298	

Character	*Character width*	*Line count remainder*	*Number of spacebands*
		(298)	
r	6	292	
s/b	4	288	9
t	6	282	
y	9	273	
p	10	263	
e	9	254	
s	9	245	
e	9	236	
t	6	230	
t	6	224	
i	5	219	
n	11	208	
g	11	197	
s/b	4	193	10
p	10	183	
r	6	177	
o	11	166	
g	11	155	
r	6	149	
a	10	139	
m	13	126	
s/b	4	122	11
a	10	112	
n	11	101	
d	9	92	
s/b	4	88	12
w	13	75	
e	9	66	
s/b	4	62	13
a	10	52	
r	6	46	
e	9	37	
s/b	4	33	14
r	6	27	
e	9	18	
a	10	8	
d	9	-1	
d	9	8	
a	10	18	
e	9	27	
r	6	33	
s/b	4	37	13

Note the broken line. At this point the counter is overset, and the program then goes into an overset routine in order to get back to the nearest word ending just short of the overset condition. Of course, it must discard the spaceband in the process. In the above illustration we have not included a character counter within each line, which might be useful for calculating letterspacing if this is to be performed by the computer program rather than by the output typesetter. After determining the last complete word that will fit on the line, the program must ascertain whether the calculation falls within the allowable interword space maximum. If it does not, the hyphenation algorithm is called for. It is only after this routine has been performed that a character count would be relevant since at this point the final end-of-line decision has not yet been made.

much (if any) of the excessive interword space should be distributed between characters as letterspacing. In the latter event, another calculation might be required—that of determining the number of *characters* in the line, so that the finest possible increments of *intra*word spacing could be distributed.

This is a line which has been set without any letterspacing.
This is a line which has been set with some letterspacing.

It may be seen, then, that the logic of the justification routine is not difficult. For the moment, we are regarding the hyphenation routine as a "black box" that is plugged into the system. In other words, it is completely independent of the logic of justification. Justification involves establishing a line count, reading in characters, keeping track of spacebands, decrementing the line counter, and determining when the counter goes negative. When this happens you back up to space out the line if this can be done. If it cannot be done, and the hyphenation routine provides a negative answer—*i.e., no hyphenation possible*—all existing programs that we are familiar with can offer no solution any better than to insert extra space (beyond the maximum limits defined) between the words in the line or between the characters in the word, or both.

When type is set by "hot metal" and the operator encounters a loose line, if he is a careful workman he may discard the preceding line or lines and try to rework those lines so as to move a word or portion of a word up or down, in order to try to solve his problem in some other way. Clearly this solution could also be tackled by computer program, but it is more elaborate than most systems people want to attempt. Moreover, some typesetting machines presently have the capability of distributing white space in such fine increments that so-called "loose lines" may cease to be offensive and may effectively be disguised.

Set width tables are vital

The vital role of set width tables should be apparent at once. Just as the justifying or counting keyboard has to access character widths so that the operator can make his own end-of-line decisions, so must the computer know the relevant width values. These must somehow be available to the computer, either because they reside in memory or can be brought into memory when they are needed.

If memory is restricted there may not be room to contain all of the tables that will be required for a particular job. (Some typesetters maintain 18 fonts or more on line and some jobs require that they all be used!) Thus it may well be that many tables are necessary. One factor that will influence the number of such tables that can be stored is the amount of room required to store each table. If each point size should have to be described by a separate table, for example, then many times more room would be required. It seems better, therefore, to store *one* table for each font—at least within a given range of sizes—and to develop a formula which takes into account the point size or set size of the type called for in a particular job. It is also extremely important to find ways to compress the font information into a minumum of space.[1]

1. Such tables may store other kinds of information, such as "kerning pair" values or ligature relationships.

Moreover, since some typesetting machines today can create type in half-point increments, the design of our system should consider the possibility that we may well want to set in 7½ or 8½ point sizes, for example, and if we have a CRT typesetter we might want to create type in 6.8 or 11.2 set. It seems desirable, furthermore, to express our unit values in terms which permit us to reduce them to a common base without incurring rounding errors. In doing so we must be able to relate to the manner in which width values are expressed by the type designer and for all the phototypesetting devices that we may wish to access. The common denominator, in a relative unit system, could be 18, 24, 36, 50, 54, or even 100 units to the em. Point sizes can be in half points or tenths of a point. The best system we have encountered accepts the desired value per em (say 100 units) and multiplies it by tenths of a point. This takes care of all possibilities except for those typesetters—fortunately very few, at the moment—which can set characters in quarter-point increments. The trouble with our scheme is that it inevitably means we may have to add up very large numbers: 100 units to the em times 60 for a six-point em yields a count of 6000 for only one em of space. But, fortunately, computers can cope with such large numbers.

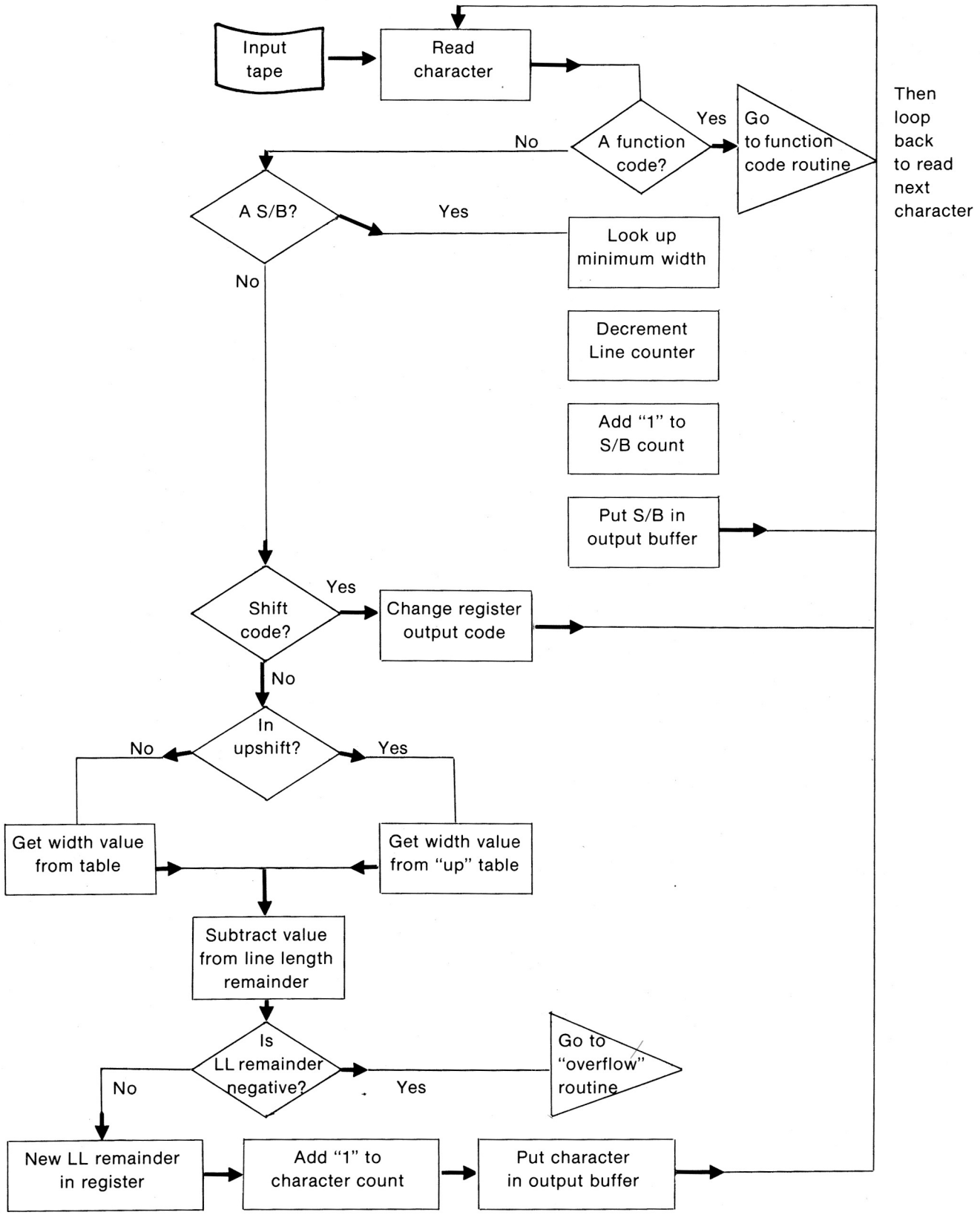
In a six-level system, if we represent widths by single characters, we will be able to describe only 64 different width values, because the six-bit configuration can yield only that many combinations. The chances are, however, that we would not need the first five or six values (one to six units, say) since they would be too low. So we could add perhaps five units to each value shown, and reach up to perhaps a 69-unit system. Nevertheless, a six-level system may have problems storing a sufficient range of values. An 8-bit code structure, giving us 256 different width combinations, would be more useful.

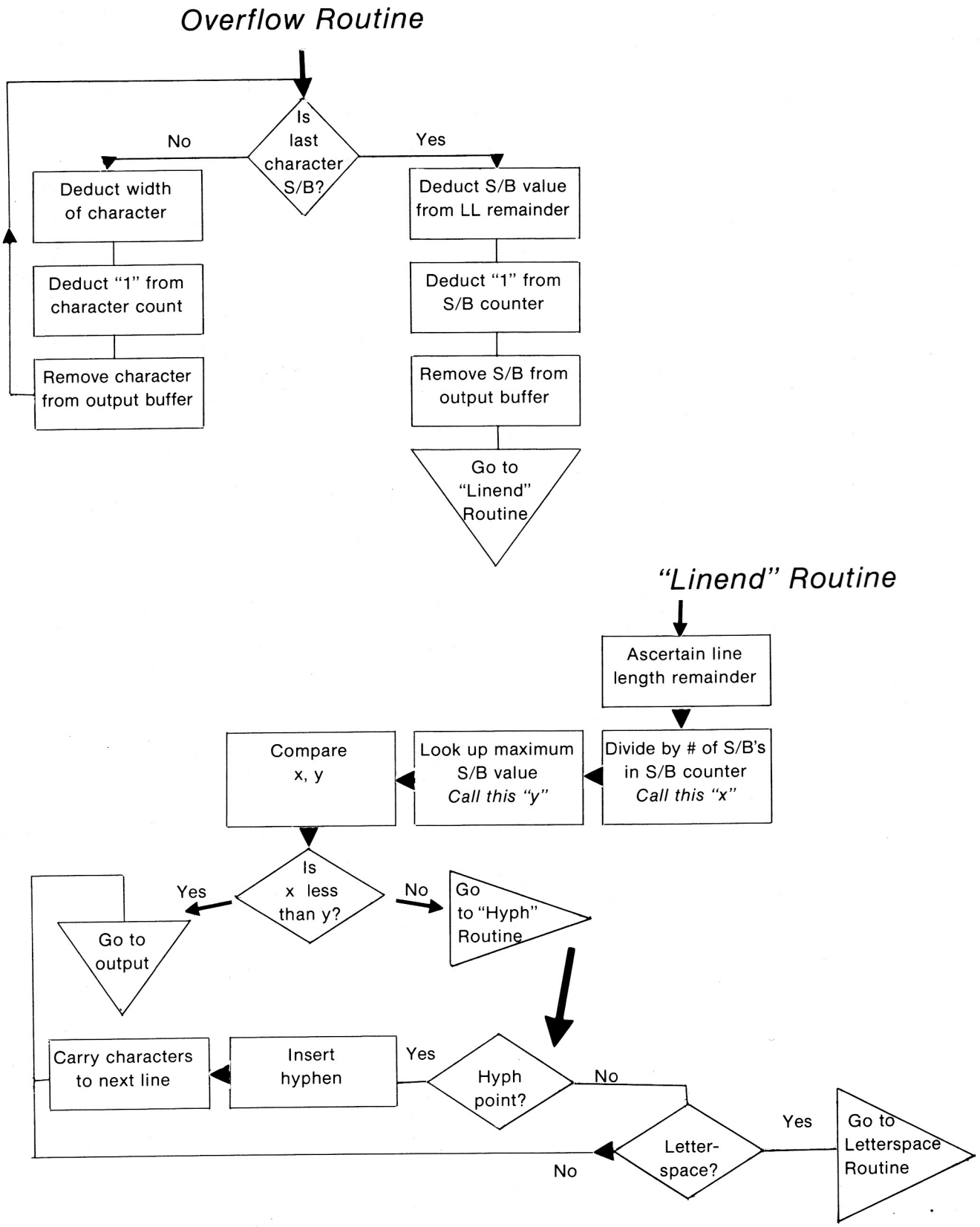
The next two pages provide a crude flow chart indicating some of the essential steps in the "justification" routine. The reader will note that we have branched to a "linend" subroutine where this is appropriate, as well as an "overflow" subroutine, but we have not shown the flow charting of the hyphenation logic, since this will be covered in the next chapter.

The underlying "justification" concept

If we think of the justification program merely as one that is designed to accept a string of text and arrange it into lines we will not do justice to the task that confronts us. Perhaps we moved too early into a discussion of justification, without sufficient attention to the fact that the underlying task is really one of *composition,* in which justification is only one aspect.

Composition goes far beyond the arrangement of text into justified lines, or even their positioning in made-up pages. Composition has to do with the aesthetic treatment of consecutive areas of white space upon which typographic and graphic images are to be superimposed. If we think of the task in this light our entire approach may well be different. We may be more concerned with the arrangement of *areas,* in terms of x/y coordinates, than with the formation of lines of text. Especially for typesetting devices like the Linotron 1010 or the Video-


Input tape
Read character
A function code?
Yes
Go to function code routine
Then loop back to read next character
No
A S/B?
Yes
Look up minimum width
No
Decrement Line counter
Add "1" to S/B count
Put S/B in output buffer
Shift code?
Yes
Change register output code
No
In upshift?
No
Yes
Get width value from table
Get width value from "up" table
Subtract value from line length remainder
Is LL remainder negative?
Go to "overflow" routine
No
Yes
New LL remainder in register
Add "1" to character count
Put character in output buffer


Overflow Routine
Is last character S/B?
No
Yes
Deduct width of character
Deduct "1" from character count
Remove character from output buffer
Deduct S/B value from LL remainder
Deduct "1" from S/B counter
Remove S/B from output buffer
Go to "Linend" Routine
"Linend" Routine
Ascertain line length remainder
Divide by # of S/B's in S/B counter
Call this "x"
Look up maximum S/B value
Call this "y"
Compare x, y
Is x less than y?
Yes
No
Go to output
Go to "Hyph" Routine
Hyph point?
Yes
No
Insert hyphen
Carry characters to next line
Letter-space?
Yes
No
Go to Letterspace Routine

Comp 570 with its full-page option, where the capability exists of producing images at any point on a two-dimensional surface of a certain size, an area composition concept will be helpful. One set of programs available for typesetting does in fact start out with the premise that text is to be "poured" into vessels or within boundaries. Others, concerned with interactive page makeup, have also adopted this approach. We shall return to it in our consideration of page makeup. Nevertheless, the problems of horizontal placement of text continue to be challenging, and we shall now look at some of them—those which go beyond the mere composition of running text, breaking copy into justified lines. What are some of the other tasks to be performed which are related to horizontal "justification"?

Quadding

Perhaps the most obvious is "quadding" textual material. Lines that are quadded are not in fact "justified" although it is still necessary for the program to go through the motions of counting the relative width of characters to make sure that they will fit within the line, and perhaps to pass this information on to the output typesetter. We must recognize, therefore, that the quad left code is not merely used to signal the end of a paragraph, but also to signal the end of the collection of words which do not—or are not intended to—overflow a single line. Sometimes, however, when there is no interactive "feedback" between the person doing the input or mark-up, it is not possible to guess precisely what will fit within one line, and how succeeding lines should be arranged. There are rules which can be applied to the use of the quadleft when it ends a running paragraph. These rules can even be incorporated into the justification program:

1. If the line almost fits its measure, space it out.
2. If the line falls short by a defined amount, use "normal"—not minimum—interword spacing.
3. If the line is very short, consisting of only a word or a portion of a word, consider the possibility of re-processing the latter portion of the paragraph (if feasible) in order to pull it up.

But none of these rules would apply to the self-contained quadded line which happens not to fit within the assigned measure:

BOROUGH COUNCIL REINSTATES WEST
CHESTER POLICE CHIEF

This suggests that "West" was re-instated as the Chester Police Chief. What the head intends is:

BOROUGH COUNCIL REINSTATES
WEST CHESTER POLICE CHIEF

Therefore we may need a different signal when we wish to quad left or—as in this case—center heads and captions. In the absence of an "interactive" program

it is very difficult to know what will fit within a line. In former times headline writers developed some crude but fairly effective methods of counting characters for purposes of fit.

The off-line keyboard operator, producing unjustified input for a computer composition system, gets no such feedback. Perhaps the markup man can continue to count out width values, or estimate them in some fashion, but this is laborious and inefficient. Fortunately, nowadays one of the capabilities of composition programs is that they can be interactive and can provide—through the video display terminal—a "window" into what decisions the computer program intends to make, so that they can be overruled.

The problem of the self-contained quadded line comes to light not only in the setting of heads and other display material, but also in the setting of poetry, or other type of lines, such as "listings" that may or may not "turn over." We therefore find it necessary to define—in the event the line does turn over—the conditions we would wish to have govern. Should hyphenation be permitted? Should the second line be indented, and if so, by how much? Are there places where the first line should not break, even without hyphenation? Should the first line be left ragged or should it be justified?

Consider some of the following examples:

Smith, G.W., 1400 Mayfield 565-2480
Society for Advancement of Barbershop Quartet Singing of America
..... 326-4777
Society for Advancement of Barbershop Quartet Singing of America
..... 326-4777

Cooper, James Fenimore, *The Last of the Mohicans*. MacMillan, 1952.
Cooper, James Fenimore, *The Last of the Mohicans*. MacMillan, 1952.

When, in disgrace with fortune and men's eyes,
I all alone beweep my outcast state,
And trouble deaf heaven with my bootless cries,
And look upon myself, and curse my fate—

When, in disgrace with fortune and men's eyes,
I all alone beweep my outcast state,
And trouble deaf heaven with my bootless cries,
And look upon myself and curse my fate—

Some of these problems may never be resolved in computerized composition unless the operator can get immediate feedback in order to appraise the aesthetic consequences of his input. But for other kinds of work—including the listings in the examples above—a general rule can be formulated and "programmed," such as to carry the runover line as a "hanging" indent, and to set the first of several such lines in an entry in a "ragged" or flush left manner if justifying it would exceed the desired interword space.

In general, as the reader well knows, the quadding function embraces such format possibilities as:

Quadding left

Quadding right

Quadding center

Quadding left and right

Leadering from left ..... to right

Leadering from left ..... to right

And generating rules _____

In like manner, runover lines can be set with the succeeding line or lines composed flush left.

In like manner, runover lines can be set with the succeeding line or lines composed flush right

In like manner, runover lines can be set with the succeeding line or lines composed quad center

Changes of measure and the use of indents

The program has to be able to cope with instructions to change line length. Such instructions are generally imbedded in the text. They may or may not occur at the beginning of the text or at the beginning of a paragraph within the text. In this instance we have given a command for the left indent to start at the beginning of the next line, and to consist of a six-pica indent. The indent will continue for six lines and then will be superseded by other commands. While we are illustrating this function on the Comp/Set and therefore, to some extent, "faking it," we are nevertheless representing what the textual presentation would look like if the condition were implemented solely by use of a computer program.

Indentions are of various types. There is the indention of *specified duration*, which is to continue for a given number of lines or points and then is to be dropped. Or there is the indention of *indefinite duration* which is to be continued until a signal is given that (as of the next line) it should be eliminated. There must be a capability of handling left, right or center indents, and to create hanging indents, and to calculate the amount of indention required to align with a string of characters whose total width the input operation does not know. This latter function if often called "indent text." Here is an example of what we have in mind in the case of an "indent text":

> The amount of the desired indent has been calculated based upon the width of the text in the preceding line. This technique is often used in the case of outline material. In this example we have had to "fake" the results using an alternative capability provided by our direct-entry typesetter.

Moreover, there must also be the capability of "nesting" indents. Here are some examples:

> We have called for a center indent, placing five picas on each side of the margin. And now within this center indent we shall begin a left indent:
>
> > This fixed space consists of ten ems of 11 points each. Note, therefore, that indents are on some occasions expressed in absolute terms, such as in points and picas, and on other occasions, in relative terms, in ems and ens. Some programs even permit stating these values in terms of inches or millimeters.
>
> Now, remaining in the same center indent, we shall insert a right, run-around indent. The left indent now remains at five picas, but the right indent is now ten picas, as might be the case if the intention were to insert some illustrative material in the space so created. There are even programs which make possible a "skewed" indent, which varies from line to line.

To illustrate the "skewed" indent we have returned to the normal measure. Now each succeeding line will be indented from the left by one additional em space. This value, could, of course, be any amount one desires, and it can skew "in" and then skew out again, or it can skew from either the right or the left margin, or from both margins simultaneously. In this instance we are illustrating a skew from the left only, and one which goes in and then goes out again. Such indents may be used to arrange text around an illustration in a "display" ad, although there are now other ways of accomplishing this, by "pouring text" within boundary lines which can be drawn on the face of a video tube.

It may be seen that a wide variety of "composition" effects may be made available either to the operator or to the mark-up person, or both. Some are quite complex, and some are simple. For example, every keyboard operator must learn how to align on numerals, and the only way to do this is to anticipate the number of numerals which may be required—whether single or double digits. The reader will remember, hopefully, that these same points were discussed very briefly in Chapter 2, especially on page 11. We repeat them here only in the context that composition "software" must be capable of offering coding commands and software routines to make these things happen automatically and not merely by manual activity. The program should also have the capability of permitting ragged setting, since this form of composition is often preferred.

Here is an example of text which is being set "justified" with occasional hyphenation. We shall then re-set this same paragraph without permitting the interword space expansion to take place. This is one form of ragged setting known as "ragged right." Then we will set the same material, line for line, also ragged, but ragged left rather than ragged right, and finally, ragged "center."

Here is an example of text which is being set "justified" with occasional hyphenation. We shall then re-set this same paragraph without permitting the interword space expansion to take place. This is one form of ragged setting known as "ragged right." Then we will set the same material, line for line, also ragged, but ragged left rather than ragged right, and finally, ragged "center."

Here is an example of text which is being set "justified" with occasional hyphenation. We shall then re-set this same paragraph without permitting the interword space expansion to take place. This is one form of ragged setting known as "ragged right." Then we will set the same material, line for line, also ragged, but ragged left rather than ragged right, and finally, ragged "center."

Here is an example of text which is being set "justified" with occasional hyphenation. We shall then re-set this same paragraph without permitting the interword space expansion to take place. This is one form of ragged setting known as "ragged right." Then we will set the same material, line for line, also ragged, but ragged left rather than ragged right, and finally, ragged "center."

It will be noticed that the differences between these samples are not very great. This is because we have set them very "tight" and thus have not allowed much room—in any event—for interword space expansion to occur. To avoid this problem, many programs for ragged setting deliberately seek to set looser, or even to require that alternate lines should differ in length from each other by as much as two ems.

Here is an example of text which is being set "justified" with occasional hyphenation. We shall then re-set this same paragraph without permitting the interword space expansion to take place. This one form of ragged setting known as "ragged right." Then we will set the same material, line for line, also ragged, but ragged left rather than ragged right, and finally, ragged "center."

A typical command structure

To give the reader a more precise understanding of the nature and extent of the composition "language" and the typographical effects that that language is intended to achieve, we shall now list all of the major commands to be found in one typical composition program. These commands are found in the IBM "Printext" software package, which, in turn, is patterned after one of the most widely copied composition programs—known as the IBM 1130 FDP software. These commands—or most of them—were used in the "Field Developed Pro-

gram" which was written by the Cleveland (Ohio) office of IBM for the model 1130 computer. Written in the late 1960s, this, in turn, was an improved version of earlier 1130 programs which had evolved from the original IBM 1620 composition software. We shall list the command codes and explain what composition effects they are intended to achieve.

AB—*(abbreviate).* To abbreviate a text string in an attempt to fit. Switch to caps and small caps from caps. Switch to caps and lower case. Drop vowels from end. Drop letters from end. *Not a common command. Not found in most software packages. Of marginal value.*

AE—*(alternate face until end).* To call for the alternate face for setting the remainder of the line only. *Here, again, this is not a common command. It was apparently devised to simplify certain kinds of input. The condition would not be encountered frequently.*

AF—*(alternate face).* To call for alternate face until further notice. *It is more usual to specify the desired type face, rather than to designate an official "alternate."*

AG—*(allow ligature).* To indicate that individually-entered characters are to be combined into ligatures. *This is a common feature of most good composition programs. It eliminates the need on the part of the operator to key ligatures, permitting the user to change his mind, and providing a text file which is more universally processible.*

AL—*(allow letterspace).* To indicate that, from this point on, letterspacing is allowed if required by the parameters which indicate otherwise an excessive interword spacing. *This command is common to virtually all typesetting programs.*

CA—*(change assembly).* To change face, using another "assembly" (*e.g., font disk*) until the end of the take or until further notice. *This is a somewhat unusual command. It presumes that the codes which designate a given font will now stand for another font, as on another disk or drum, and it arises out of the limitations of some typesetters. But it is unusual to change a font "dressing" in such cases during the course of a job.*

CC—*(change column).* To specify the column, or more precisely, the line length or measure to be used to set the following text. *This command or its equivalent is common to all typesetting programs, and it must be given at the beginning of every take (if not more frequently) unless "default" conditions are to be observed.*

CF—*(change face).* To specify the face (font master) to be used for setting the following text. *Like the command immediately preceding it, it is vital for all typesetting programs.*

CL—*(change leading).* To specify the amount of leading to be used to set the following text. *Again, this is a common command, always given unless a default condition is expected to prevail.*

CM—*(compare strings).* To compare the contents of two strings and to set a result. *This is an unusually sophisticated typesetting command not commonly found in composition programs. Its purpose is to permit the user to cause text to be set differently depending upon what conditions are encountered. Conditional commands of this character are found only in quite sophisticated software packages, although the same results may be achieved in other ways, especially if the program is "interactive."*

CO—*(comment).* To store information with text for later use, as for a user-provided post-processing program or to include information relevant for a subsequent operation. *Many composition programs offer the user the opportunity to bury editorial comment or instructions within the text, which the composition program will ignore, and not typeset.*

CP—*(change point size).* To specify the point size to be used for the following text. *This is of course a vital command, always given at the beginning of a job, and frequently during the course of a job. The syntax of the command is usually something like [CP06] for a change to six-point type.*

CS—*(change set size).* To indicate that the following text is to be set in a different set size. *Some programs assume the same set size as the indicated point size, but increasingly this is no longer true. However, there may be a default rule in this connection. Where this command is given it requires that the program recalculate width values to the new set size, just as would be the case if a point size command change were given—implying a change in character widths.*

CT—*(change table).* To request a different keyboard or printer graphics table. *This assumes that there are various "overlays" of graphic symbols which can be substituted one for another, either on input or output, or both.*

DE—*(do end).* To define the end of a series of commands that are controlled by a **DW** or **DI** command. *(See below.)*

DF—*(define function).* To specify data to a user-written routine to provide implementation for unique features of the device. *This type of command is becoming increasingly common. It provides for coding that is "passed through" the program without any need for processing, since it expects the output device to cope with it. It might be a signal for the CRT typesetter to draw a vertical rule of a certain depth at a certain location.*

DI—*(do if).* To conditionally execute a series of composition commands. *Again, this is a sophisticated and not-too-common command of more value in a "batch" environment than in an interactive one where the user could intervene and change the conditions himself.*

DL—*(define leader).* To define the character to be used for leader insertion. *While this command is quite common, it is also common to have the ability to define two such characters and to alternate them in a certain order, as with a dot and a dash.*

DM—*(define module).* To indicate a job table that specifies a device module number to be used and/or an output set-up number. *Many programs now make it possible, one way or another, to process text which is destined for one of several (perhaps on-line) typesetters.*

DO—*(define operator).* To define the operator who input or edited a job so that statistics with respect to operator performance can be developed and credited to the operator. *This type of management control function is increasingly common as an "add-on" to composition software.*

DW—*(do while).* To loop through a series of composition commands while the specified item is positive. *Again, this is a conditional command perhaps more appropriate for a batch program. It is not common in other software programs at this time. Of course, the condition that governs must be specified by the command "argument."*

EF—*(end format).* Specifies that the format statement is being defined in order to be terminated and stored. Also used to denote end of execution when format is called. *To understand this, the reader will have to refer to our discussion of "stored formats" which follows.*

EL—*(extra lead).* To specify extra or reverse (minus extra) leading for immediate use, or to set (or cancel) an automatic leading mode. *Extra leading helps to space out heads. Reverse leading enables the operator to position special characters off (above) the base line, or to lay down a second column alongside a first.*

EP—*(end paragraph).* To specify that the present paragraph is to be immediately terminated. *A quad left code would do just as well.*

ER—*(erase line).* To prevent the current line from being output to a photocomposer or output device. *This command is not relevant for a composition program, but is concerned with what to do with erroneous input. However, there are some other "conditional format" circumstances in which it might be employed.*

ES—*(end of short).* To separate short takes when it is desired to reset the system defaults before each part.

ET—*(end of take).* To indicate that the current take is to be terminated.

EX—*(extract).* To move data from an input record to a numbered string for data-processing-related activities. *Not a common composition command.*

FA—*(font assignments).* To specify the physical fonts available and their locations. *This is generally not part of a composition package but is rather a "pre-processor" "utility" routine to enable the software to address the desired font locations in order to access designated characters.*

FC—*(film cut).* To cause film or paper to be cut by the output typesetter into a desired length or take.

FS—*(fixed space).* To cause the immediate insertion of a fixed amount of space in a line. *The "argument" following the command will indicate the value of the fixed space, usually in relative units.*

GE—*(get at event).* To indicate a format to be invoked when certain conditions occur, as if a "get format" command were issued. *This is another of the more sophisticated "if" commands not commonly included in run-of-the-mill composition programs.*

GF—*(get format).* To call from disc, in "get" mode, a previously defined format. *Get and use format commands are basic to virtually all composition programs. But sometimes such formats are stored in memory rather than on disc.*

GS—*(get string).* To compose the contents of a string. *Similar to a "get format" except that the string could be text or a combination of text and format instructions.*

HO—*(homograph).* To specify where hyphens are to appear in a word which is a homograph—that is, a word which hyphenates differently depending upon meaning. *A discretionary hyphen will accomplish the same result.*

HT—*(hyphenation test).* To hyphenate a word or words and print out all the hyphen points produced. *This is usually a feature of a "utility" program rather than the basic composition program itself.*

IF—*(indent first line).* To specifiy the amount of left or right indention of the first line of all succeeding paragraphs *(ascertained by test following a quad left, for example). This command obviates the need for keying an "em" indent at the beginning of each paragraph.*

IH—*(indent hang).* To specify the amount of left and/or right indention of all lines of following paragraphs *other* than the first line. *The argument must of course indicate the value of the indent, as in ems and ens, picas and points, or even fractions of an inch or millimeter, depending upon the program.*

IL—*(indent left).* To specify an amount of left indention for a given vertical distance. The duration of the indent may be indicated in the argument—as so many lines, or so much depth in points—or it may continue until cancelled.

IM—*(imbed take).* To switch the input stream to the referenced take and include it at this point. *Not a common software feature.*

IN—*(indent).* To set indention values and cause indention. *More commonly a left, right, center or hanging indent is specified.*

IP—*(indent paragraph).* To specify the value of left and/or right indent for all the following lines in the current paragraph only.

IR—*(indent right).* To specify amount of right indention for a given vertical distance or until further notice.

IT—*(indent take).* To specify the amount of left or right indent for all the following lines in the current take only.

IX—*(indent text).* To specify an amount of indention when the numeric value is unknown but the indention is based upon the positioning of the command within the line.

JU—*(justify line).* To terminate the current line at the present point and to seek to justify it whether or not it accords with normal justification parameters.

LI—*(leader insert).* To specify the location of leaders for lines ended by a quad, from the point of the command or from a "quad middle" command, or for tabbed lines.

LJ—*(line justify to fit).* To specify that a line of type is to be set, using one or more system-selected point sizes, so that the line fits within the column measure at the largest size possible.

MC—*(merge control).* To transfer program control between the format input stream and the normal input stream.

MT—*(measure text).* To output an undeterminable amount of space or to save the value of this space as a variable. *This is a most useful command to enable the user to ascertain how much space is taken up so that he can issue commands to reverse lead or end a page. It approaches the pagination routines which will be discussed in a subsequent chapter.*

NF—*(normal face).* To signify that the normal or designated face is to be used from this point on until further notice. *It may save the operator the problem of finding out what the normal face is supposed to be. Not a common command.*

NS—*(non-space).* To request that the next character be set with zero width, as to permit the positioning of a "floating accent."

OF—*(off switch).* To set the switches specified to the "off" position. *In this context a "switch" is a routine to be invoked. This is not a common command for composition markup.*

ON—*(on switch).* To set specified switch to "on."

PS—*(place string).* To clear and copy a single string or to concatenate multiple strings. *Again, not a common command since it is more sophisticated than what is offered in most programs.*

QC—*(quad center).* To indicate that the present line should be immediately terminated and quadded center.

QF—*(quad from middle).* To terminate a line in which leaders or white space is to appear. Multiple leader insertions (**LI**) and/or white space (**WS**) may appear in the line. *In other programs it is sufficient to terminate the second half of the line with any quad code.*

QL—*(quad left).* To terminate the current line and quad it left after any left indention is placed.

QR—*(quad right).* To terminate the current line and quad it right after any right indention is placed.

RC—*(ragged center).* To produce ragged left and right margins.

RD—*(read record).* To open and read an external file. *Not a common command, since it is concerned with accessing files by a VDT. Interactive video-oriented programs offer various techniques for getting at such files.*

RI—*(remove indent).* To remove all, or part of the right margin indent of the current line.

RL—*(ragged left).* To produce a ragged left margin.

RN—*(repeat number).* To set multiple copies of the same line. *Not a common command.*

RR—*(ragged right).* To produce a ragged right margin.

SC—*(set conditional).* To set a variable with one of three values depending on the sign or zero condition of a specified data field. *Not a common command—implying much more sophistication than most programs offer.*

SF—*(save format).* To save text and/or commands on disc under the format number specified. *Most programs offer this capability, except that the "saving" could be done in memory.*

SI—*(set item).* To cause up to five digits, representing the integer value of an item, to be set in the input stream. *Unique to this particular program.*

SL—*(skew left).* To create copy with skewed indent on the left side.

SN—*(slant).* To specify to the output device that the text which follows is to be set in a slanted (obliqued) mode. *Usually a "pass through" command for CRT typesetters.*

SR—*(skew right).* To create copy with a skewed indent on the right side.

SS—*(substring).* To create a new string from a portion of another string. *More or less unique to this particular software.*

ST—*(save text).* To place text into a format where everything else is standard. Used in conjunction with **get at event** command, it allows the keying of copy in one sequence and setting in another. *Unique to this particular software package.*

SV—*(set variables).* To set the values of variables to be used in later commands. *Not a common feature of composition software.*

TB—*(tab between).* To specify the end of tabular material and cause the text to be centered between the next two available stops.

TC—*(tab center).* To specify the end of tabular material and cause the center of the text to be placed at the next available tab stop.

TJ—*(tab jump).* To specify a skip to the tab stop specified. *Not common.*

TL—*(tab left).* To cause the word preceding the command to be tabbed flush left against the next tab stop. Or to indicate the beginning of the first text in a line which is going to be followed by a tab command.

TN—*(tab number).* To set a specified number of equally spaced tab stops.

TP—*(tab proportion).* To set tabs when the exact location is not known but the proportional distance between them is known.

TR—*(tab right).* To specify the end of tabular material and cause the text to be tabbed right flush against the next available tab stop.

TS—*(tab set).* To set tabs at the points specified. *It will be seen that this program offers a great deal of tab setting sophistication.*

TW—*(tab width).* To set tabs at the point specified in the first data field plus the widths specified in the following fields.

TX—*(tab text).* To specify a tab point when the numeric amount is unknown.

UF—*(use format).* To call from disc a previously defined format statement.

UN—*(unslant).* To specify to the output device to set the following text in a normal mode.

VC—*(values for composition).* To set one or more values to control aspects of composition. Column or line width. Width on the left side into which punctuation may be placed. Width of margin on right side into which punctuation ("hung punctuation") may be placed. Number of desired hyphenation module (1 to 10). Number of exception word dictionary to be used, or no dictionary. *While many of these are unique the effort to achieve quality composition such as "hung punctuation" or to avoid multiple hyphens, or get a better quality of hyphenation, is common to several outstanding programs.*

VH—*(values for hyphenation).* To override temporarily the default hyphenation values in the job table.

VL—*(values for letterspacing).* To change temporarily the default letterspacing to be applied.

VR—*(values for ragged copy).* To override temporarily default values specified in job table for ragged copy.

WS—*(white space).* To cause available room in a line to be placed at the white space command indicator.

XG—*(cancel ligatures).* Ligatures should no longer be combined.

XH—*(cancel hyphenation).* Hyphenation is no longer to be done.

XL—*(cancel letterspacing).* Letterspacing is no longer to be done.

XR—*(cancel ragged).* Return to normal margins, setting all lines flush left and right.

It will be seen that most of the above commands are "mnemonic"—that is, the "name" of the operator suggests its function, to help the markup person remember what the commands stand for.

During the course of the explanation of these commands, we have alluded to the fact that many of them require "arguments"—that is, the amount of duration of an indention, or the size of the type font, or the number of the font, must be specified as part of the command. Obviously many of these commands are quite complex, and they must be input quite precisely or else the program will ignore them, or become confused and not know what to do. Thus an effort has been made to simplify them. Unfortunately, this is not easy to do, and the best that can be achieved—as a rule—is to identify those which are repetitive and to be certain that they are marked properly and keyed and input correctly, and then to store them away as formats which can be called out with one or two unique keystrokes much easier to keyboard. Commands, unfortunately, are difficult to input exactly because they are not familiar words and do not come easily to the keyboarder's fingers. "String substitution" helps to overcome this problem, and one simple, unique string can be used to stand for not just one command, but for several

commands which are invariably used together. Hence most composition programs now offer the ability to substitute for a simple string of codes a more complex string. These can be stored away as formats, and designated by such commands as "Use format 01"—[*UF01*]—or some other kind of "string substitution" routine can be invoked. The unique combination "$a" can be exploded into a very complex set of instructions, or a combination of instructions and text.

It is often troublesome to develop the proper codes to achieve desired results, but once they are "debugged" by testing, and stored away, thereafter their use is quite trouble-free, and ordinary keyboarders can be taught to produce composition results which are quite complex.

The use of stored formats

Stored formats are therefore the salvation of computerized photocomposition. Modern phototypesetters offer many capabilities, but to cause these machines to go through their paces and thus to enjoy all of their benefits is usually quite difficult. Hence, stored formats are called into use. Not only can such formats be exceptionally complex in and of themselves, but they can also call out other formats. In other words, stored formats can be "nested" and this is particularly useful if the program in some way limits the length of a given format. By tacking on to the end of that format still another one, you can get the benefit of a more complex statement of conditions. There are also formats which are called "merge" formats, and which contemplate that one will, at a given point, "exit" from the format, and at a given signal, usually known as a "merge" call, return "control" to the next step within the format again. On the right is an example from the DEC (Digital Equipment Corporation) *Typeset-11* composition software manual. In this instance the commands are expressed differently, but they accomplish the same results. Within brackets a "combined command" is given. Thus, "f2" means "font 2," "v36" means vertical leading of 36 points, and "c10.6" means a column width of 10 picas and six points. By studying this example you will see how the "at" sign @ is used to return to the format again and again.



1. *Straight markup*

```
[f2p27v36c10.6]          SHOES FOR/c
[v24]                    SUMMER/c
[f6p12v16]               Our Reg. . . ./c
[v12]                    4 Days Only/c
[f4p72v72]               2 95/c
[f1p8v8va12]   All brand new . . . 5-9./1
```

2. *Format definition*

The definition is written simply by copying everything over from straight markup except, in place of each block of text, enter a call point @ as follows:

```
*DEPTAD1 = [f2p27v36c10.6] @
/c[v24]@
/c[f6p12v16]@
/c[v12]@
/c[f4p72v72]@
/c[f1p8v8va12]@
/1 =
```

3. *Format use*

```
*DEPTAD1*SHOES FOR@SUMMER@
Our Reg . . .@4 Days Only@2 95@
All brand new . . . .@
```

Note: When a block of similar ads has been marked up, the format should be cancelled to avoid unwanted invocation by an erroneous @ in original input. Hence, if "SHOES FOR SUMMER" was the last ad (example above), the markup would conclude:

$2.95@All brand new . . . .@*%

Summary

The "justification" process therefore embraces a wide variety of composition routines, many of which are extremely detailed and all of which are meticulously conceived to achieve the same capabilities which a highly-skilled craftsman had learned to give effect to in hot metal. With the use of these commands and the routines which they call forth—branching, as the reader will recall from our earlier discussion, away from the character-by-character processing loop to the additional steps and complications implied by the composition language we have sought to illustrate.

We still need to know more about the environment in which this activity takes place, but first it will be helpful to gain some understanding of one other aspect of the composition task, and that is hyphenation, or the splitting of words —not by human decision, but by computer algorithm or by dictionary look-up, or a combination of the two. Then we can return to a more detailed examination of the "system" aspects, in which the composition program is performed, including the manner in which the user can interact with the "system" to make certain that his input is correct, and that his output will be formatted and processed in the manner he has visualized.

Review Questions:

1. Why are tables of width values important to the composition process?
2. Describe the way in which a determination is made as to how many characters and words will fit on a line.
3. What information must the program have readily available to make its calculations? How does it get this information, and when?
4. Identify a number of subroutines within the main composition program that the program will be compelled to "branch to" and describe what results these subroutines are attempting to achieve.
5. While we have not discussed tabular composition in this chapter, except to indicate what some of the commands are, try to visualize how a tabular program would be written, and what problems would be encountered in defining the task.
6. Try to figure out how a program could be devised to "hang punctuation."
7. What is the difference between a regular left indent and an "indent text" command, as we have described it? What special problem does the "indent text" command have to cope with?
8. What would the program have to "know" in order to calculate a left indent which would last for twelve lines, and then would automatically be cancelled? How could the program find out this information?
9. What do we mean by "string substitution"? What are "stored formats" and how would they be used? When would it be advantageous to use them and when would their use be to no avail?

14

Hyphenation Examined

A hyphenation routine is an essential part of a justification program. Without hyphenation, the quality of justification is unacceptable except perhaps in very wide measure composition. There are techniques for "hyphenless justification," but without some recourse to letterspacing the results are such that one might better consider ragged setting as a more pleasing alternative. (Even in ragged setting some hyphenation is desirable to conserve space and to provide for a more pleasing appearance.) Here are two examples, in narrow measure, of hyphenless justification, the only difference between them being in the presence or absence of letterspacing.

Justification and hyphenation go together. There is an intimate relationship between these two concepts. The greater the latitude afforded for spaceband expansion the less frequently hyphenation is required. If there is very little allowance permitted for the expansion of spaces between words, then hyphenation must be performed very often. The first criterion for successful typesetting by computer composition, therefore, is to work out, for a given assignment—considering the desired measure, type face, point size and the presence or absence of large words—the proper assignment of minimum and maximum spaceband values.

Justification and hyphenation go together. There is an intimate relationship between these two concepts. The greater the latitude afforded for spaceband expansion the less frequently hyphenation is required. If there is very little allowance permitted for the expansion of spaces between words, then hyphenation must be performed very often. The first criterion for successful typesetting by computer composition, therefore, is to work out, for a given assignment—considering the desired measure, type face, point size and the presence or absence of large words—the proper assignment of minimum and maximum spaceband values.

It seems clear enough that if margins are to be justified, hyphenation is an essential exercise in order to avoid awkward and unseemly intra- and interword spacing. To the extent, then, that valid hyphenation points cannot be discovered by the program, and no hyphenation occurs, the quality of justification suffers.

It is also evident that the frequency of hyphenation, within a given context, will be, among other things, a function of the amount of interword spaceband expansion which is specified. Here are examples of the same text set with different parameters for interword spacing, resulting in more or less frequent hyphenation and successively looser settings, with obvious impact upon the quality of justification.

Justification and hyphenation go together. There is an intimate relationship between these two concepts. The greater the latitude afforded for spaceband expansion the less frequently hyphenation is required. If there is very little allowance permitted for the expansion of spaces between words, then hyphenation must be performed very often. The first criterion for successful typesetting by computer composition, therefore, is to work out, for a given assignment—considering the desired measure, type face, point size and the presence or absence of large words—the proper assignment of minimum and maximum spaceband values.

Justification and hyphenation go together. There is an intimate relationship between these two concepts. The greater the latitude afforded for spaceband expansion the less frequently hyphenation is required. If there is very little allowance permitted for the expansion of spaces between words, then hyphenation must be performed very often. The first criterion for successful typesetting by computer composition, therefore, is to work out, for a given assignment—considering the desired measure, type face, point size and the presence or absence of large words—the proper assignment of minimum and maximum spaceband values.

Justification and hyphenation go together. There is an intimate relationship between these two concepts. The greater the latitude afforded for spaceband expansion the less frequently hyphenation is required. If there is very little allowance permitted for the expansion of spaces between words, then hyphenation must be performed very often. The first criterion for successful typesetting by computer composition, therefore, is to work out, for a given assignment—considering the desired measure, type face, point size and the presence or absence of large words—the proper assignment of minimum and maximum spaceband values.

But the fact that the program calls for hyphenation does not assure that the words submitted to the hyphenation routine will in fact be divided. These are words which are encountered at the "fall-off" point—that is, where their inclusion is impossible, but their exclusion would necessitate overexpansion of the spacebands beyond the amount stipulated. The writer often uses the term "straddle words" to identify those so situated. Clearly not all straddle words should or could be hyphenated.

What constitutes a "word"?

The first question to be answered is what *is* a word for purposes of hyphenation? The first step must be to identify a certain string of characters as probably embracing a word, based upon their "delimiters"—that is, the unique characters surrounding the string. Obviously a spaceband on either side of the string would enable us to determine that there may be a word in between. But not only spacebands serve this function. For our purposes, a hyphen—as in a compound word—provides a similar delimiter, whether it occurs at the beginning or the end of the string: micro-computer and multi-purpose are examples. We might not want to hyphenate one of the components of a compound word string, but the component would be a word nonetheless. In like manner, an em dash or an en or "nut" dash could serve as a delimiter—as you can see right at this point. "Delimiter" and "as" are words in their own right. A fixed space may also be considered a word delimiter—as with an em space at the beginning of a paragraph, and a quad, such as a quad left, would serve as a delimiter as at the end of a paragraph.

Another more troublesome delimiter may be the "slash" when it is preceded or followed by "alpha" characters. Hence, *and/or* would be two words, rather than one. And in scientific terminology one frequently encounters similar expressions. Even in our own vocabulary we might choose to use an expression such as the "word-processing/typesetting interface."

"Cleaning up" the word

Once a possible word has been identified, we must examine the string more carefully to make certain that it consists of mostly "alpha" characters. The hyphenation routine is only concerned with the 26 letters of the alphabet, and must temporarily set aside other elements that might be found within this string. It may be, for example, that we are dealing with numbers. "There are 1001 ways to approach this problem," one might say. But "1001"—although delineated by word spaces—is not a hyphenable word. On the other hand, it may be permissible to split numbers, if they are long enough. If we write out a number string such as 1,000,000,000, it may be necessary to split it, being certain that we leave behind or carry forward at least three digits. But this string, when so analyzed, would not normally be referred to a "hyphenation routine." Rather it would go to a separate number-splitting routine.

Moreover, there are some combinations of words that should not be divided even though they may be (or seem to be) separated by spaces. It may be convenient to use an "en" space to provide such separation, and not to identify it as a word delimiter. We have in mind the following situations: 345 mi., where "mi." belongs with the preceding number. Or 24 kg. or 55 B.C., or 6:45 P.M. Or, perhaps, an expression such as "O Lord." When a nut space is so used and not regarded as a delimiter, the line will sometimes look rather strange, if that nut space happens to have a conspicuously wider or narrower value than other spaces within the line. Consequently, we have often advised composition systems' designers to provide a unique character which does not serve as a delimiter, but is considered as a space in that it takes on the value of other spaces within the line.

Let us return, however, to the subject of "cleaning up" the word. If it is a word at the end of a sentence, allowance must be made for the fact that to the left of the next interword space or quad left, there will be a period. But it could be something else. It could be any one of the following, and some or all of these conditions might apply with respect to words anywhere within a sentence:

computer."
hurry!
forever?
illustration.[1]
chart*

Or, there may be something at the beginning of the word, such as:

"However,
•Note
©reserved

And there can even be "garbage" in the middle of the word. This is invariably the case when the word contains an accent or a diphthong. But there may be coding to provide for a font shift, such as to give emphasis: "You must be mis*taken!*" It is even possible for the "garbage" to consist of "alpha" characters, such as an alphabetic footnote call-out reference.

Differentiation of capital letters

Even before going to a hyphenation routine, there are some programs that scan the "cleaned-up" word to see if it contains any upper-case characters. (We are assuming that if the word consisted of *all* upper-case letters it would hyphenate normally, and that this would be true if it consisted of caps and small caps.) But if a capital letter occurs in the middle of a word it may be permissible to split before this capital. This would most likely be the case with proper names, such as McLean, MacIntyre or MacKenzie. They are not always Scottish, since one occasionally encounters a "duPont" or a "benZion," a "vonBitter," or a "delaRue." There is, however, a rule never to divide proper names, but this is often—if not frequently—impossible.

How big a word must it be?

Let us assume that we have identified a word string, and that we have "cleaned up" the word by removing all non-alpha characters, setting them aside since we know that they must be put back into the string, and proper allowance made for their widths. We must also remember, in our program, to be sure that if the divided word requires a hyphen, there will also be room to add the hyphen at the end of the line. What we have left should consist of more than two letters. Some routines permit the user to specify how many letters the word must contain before it is to be sent to the hyphenation algorithm. The accepted rule is that the word must contain no less than five letters, but there are some users who are willing to divide four-letter words, and perhaps there are some who would not want to split a word unless it contained six or more characters. Our own preference is the five-or-more letter combination, with the understanding that generally one would never "leave behind" fewer than two letters. Hence, one would not hyphenate a-gain, or a-head, or u-nion.

There is another rule, more honored in its breach, which suggests that not fewer than three letters should be carried forward. Therefore "may-be" should not be divided. This rule is valid for wide measure setting, but is hardly enforceable in either a newspaper or news magazine environment.

Vowel group analysis

Now we come to the logic of the hyphenation algorithm itself. We have discovered the word, we have "cleaned it up" so that it is no longer a "dirty" word, and we have counted the characters in the clean word to see if there are at least five. It is time to go into the logic of the hyphenation routine.

Some programs first scan the word to satisfy themselves that there are at least two "vowel groups." Others prefer to do the vowel group analysis at the end of the routine to be certain that there is at least one vowel in the portion of the word left behind and at least one vowel in the portion of the word carried forward. In the English language vowels are *a, e, i, o* and u, and sometimes y! It is the *y*, of course, that is troublesome. The "y" in "yesterday" is not a vowel. The "y" in

"merely" is. But it is the "y's" in the middle of the word that cause problems. Consider, for example, "dystrophy," and "leapyear."

Every vowel does not constitute a vowel group. "Through" and "straight" contain only one vowel group, so we can see that vowels must be separated by one or more consonants to be considered as within a separate group. However, not every vowel so separated can be treated as a vowel group. There is the problem of the "silent *e*." The "e" is silent in "moose" or "proves," but not necessarily in "rages" and certainly not in "ragged," although it *is* silent in "proved." Most hyphenation routines that make use of a vowel group analysis—and all should, at some stage, if only to check the results—will assume that an "e" at the end of a word, unless preceded by an "l," is to be disregarded for purposes of vowel group analysis. And many routines will also apply the same rule if the letter "e" is the second-to-last character in the word, followed, perhaps, by an "s" or a "d."

But perhaps the problem of determining the number of vowel groups is not mandatory—at least in the early stages of analysis. Perhaps this problem can be dealt with in a string analysis of alpha characters, so that at this point our only problem could be to determine that the word *may* be hyphenable. However, at some place in the routine this difficulty will be encountered and must be resolved.

What hyphenation programs do

In general, hyphenation programs consist of one or both of two kinds of procedures. One is a "logical" approach. The other is a "dictionary." Let us consider the *logical* solution first. As a rule such algorithms do not examine the *entire* word, but only inspect a portion of it. In fact, one of the most common algorithmic tests is to ascertain whether a hyphen may or may not be placed between the second and third of a string of four characters. The analysis may not begin at the back end of the word, or the front either, for that matter. The program may start with a group of four characters which include the last two which, with the value of a hyphen, would barely fit on the line, and add to the string the next two "fall-off" characters. If the word which straddles the fall-off point is "photographer," and the last two letters that could be included within the line are "gr," then there would appear to be no purpose in testing more than the two letters beyond this location. Hence, the first interrogation would be made of the tetrad:

photo(gr/ap)her

Three questions might be asked:

1. What are the possibilities of a hyphen occurring *before* "ap"? The answer might be that the chances are relatively high.
2. What are the possibilities of a hyphen occurring *between* "r" and "a"? Again, the possibilities might be good.
3. What are the possibilities of a hyphen occurring *after* "gr." The answer would most certainly be negative.

The logic program, being based upon an analysis of the English language, should be able to ascertain that no syllable can end with a "gr." Hence, even

though the answer to the first two questions was not negative, the hyphen point fails to meet the test. Hence this hyphenation point would be rejected and the program would then shift one character to the left, and look at:

phot(og/ra)pher

Here, presumably, we should get an affirmative answer to all three questions. This hyphenation point would be accepted. Note that the program would never look at the entire word. But even if it did, it would confine its attention only to each four-letter combination, and, at least in this example, the answers would provide more affirmative points than would be valid. If the analysis shifted one more character to the left, and looked at:

pho(to/gr)apher

the answer would again be affirmative. While this solution would be valid for "photographic," it would not be proper for "photographer." Hence, a correct solution to this and many other word breaks would be a matter of chance, depending upon what particular combination of characters would fit within the line.

There are some hyphenation programs which start at the beginning of the word rather than from the back end, and take the first hyphenation point that the routine encounters and verifies—stopping there without proceeding any further to determine whether or not another answer which provides for a tighter line might not be found. If the program is instructed to take the first acceptable hyphenation point "and run" it would certainly be better to work left from the fall off point rather than from the beginning of the word.

Although it requires more processing time, it seems better for the program to examine the entire word—even that portion which would not possibly fit—to feed back all the answers to the justification program, and let that program select the place which most nearly satisfies the justification needs of the particular line. In this connection, some weight might be given to hyphen points that appear more favorable than others. Statistics might be calculated as to the relative probability of a successful solution, perhaps somewhat in the following manner:

Tetrad	*Probability*	*Tetrad*	*Probability*
ph/er	90	to/gr	85
ap/he	70	ot/og	60
ra/ph	85	hp/to	80
gr/ap	0	ph/ot	60
og/ra	80		

In this case, the first three solutions would be discarded because they would not fit within the line. The highest probability solution remaining would be "to/gr" —which, unfortunately, would be wrong.

Where would such statistics come from? Many researchers in the field of hyphenation have input large dictionaries, along with their hyphen points, and have developed computer programs to study the probabilities of hyphens falling

before or after certain letter combinations. One drawback to such research is that the raw data has often been unreliable. The "input" has consisted of too few words, words have not been weighted in accordance with the frequency of their use, so that uncommon words have been given as much consideration as those used very frequently, and syllabification points have been used, rather than genuine hyphenation positions. But even if these weaknesses had been corrected, we suspect that the statistics so generated would not produce results any better than those one could develop through the application of his own common sense.

Our illustration has already prejudiced our analysis of this type of hyphenation routine, but we chose a difficult example! Whether you inspect the whole word or only a portion of it, if the answer is found by looking at only a portion there is no assurance, other than coincidence, that the right answer will be achieved consistently. Of course, there are certain sure-fire combinations, but these, unfortunately, are rather rare. For example, one can be fairly certain that:

a syllable will not end with:	*a syllable will not begin with:*
bc, bd, bg, hb, bj, bk, bm, bn, bp, bq, br, bv, bw, bx	bb, bc, bf, bg, bh, bk, bm, bn, bq, bs, bt, bv

If we take just these cases of consonants beginning with *b* (and our list should be extended to consider *all* consonants) we find hyphen points after the first category and before the second would be prohibited. Notice that this particular test is useful (if our empirical observations are correct) not for the purpose of finding a hyphen point, but for the purpose of rejecting one found by other techniques. One program we know of makes all of its hyphenation judgments solely by an analysis of vowels and consonants (without any regard to what specific ones they are!) Once such a determination is made, a second check of the kind indicated here would certainly be helpful!

It is not, in fact, very difficult to find acceptable syllable breaks by studying combinations of two or more consonants. There are, of course, certain problem combinations. Where two *b*'s occur together the chances are greater that the hyphen would appear *between* them, but it could occur *after*. Where *bbt* occurs we can be certain that the hyphen would follow the second *b*, as in *Crabbtree, but where sth* occurs, the word could be *masthead, asthenia,* or *asthma.* The *st* combination is particularly difficult, as witness as/ter, cast/er, and be/stir.

Hyphenation procedures are much more complicated in those cases where we are dealing with the occurrence of a single consonant, with a vowel on either side. For example, consider bas/il, ba/silic, ba/sin, and bas/inet. On the whole, there is probably only a 50% chance of being correct when dealing with an otherwise unidentified portion of a word containing a single consonant at the proposed hyphenation point, such as:

re/si	re/side, res/i/dence
re/se	re/serve, pres/ence
em/or	fem/oral, de/moralize

The accuracy of the algorithm may be enhanced somewhat by taking a five or six letter combination, such as:

de/mor or dem/ora

but it is doubtful whether the inclusion of one or two additional characters in the diagnostic string will contribute significantly to the solution of the problem.

Word structure analysis

Another approach to an algorithm involves a more substantive analysis of the structure of the word itself, aimed at identifying prefixes and suffixes as well as "root" words, and treating each component independently. As a general rule, for example, it is permissible to hyphenate after a prefix, such as *re* in *reentry* or *restructure.* Another example might be *dis,* or *mis, in* or *un,* or perhaps *trans.* Suffixes such as *er, tion, sion, cious, tious, ceous, able, ible, ness, ish, ate, ive,* and *some* may likewise be acceptable post-hyphen syllables in many cases.

Looking first at the prefix possibilities, we may generalize that a hyphen may almost always be placed after *in* at the beginning of the word, but there are a few exceptions, such as *inchworm, inkless,* and *innkeeper.* The prefix *un* would give us trouble in *union, unction, unicorn, unified, unison, united,* and *universe.* The prefix *dis* wouldn't work for *dishwasher,* and *mis* would cause difficulties with *misogyny,* and *mishmash. De* would bring trouble in the case of *decalogue, debonair, debit, debutante, decadence, and many others. Anti, as a prefix, would produce the wrong result for antiphony* and *antiquity. Re* would be most unsuccessful. Witness *reappear* and reaper, react and *ready, recognizance,* and *recognize. Inter* will present problems for *internship* or *interior.*

Looking at suffixes, there are *dream/er* and *boul/der, check/er* and *chat/ter, chas/er* and *Chau/cer, control/ler* and *poll/er, floun/der* and *bound/er.* Or consider *ca/pable* and *wash/able, commu/nism* and *social/ism, patron/age* and *out/rage, medi/cable* and *limit/able, contempt/ible* and *admis/sible, consum/mate* and *elector/ate, admon/ish* and *bran/dish.* Clearly, there are relatively few trouble-free suffixes. Probably *tion* and *some* and *ness* are the most consistent. The letters *ing* are unusually difficult, for we have such peculiarities as let/ting and grat/ing, grasp/ing and shop/ping, stall/ing and appal/ling, whis/tling, whisper/ing and confer/ring.

Of course, many of the apparent contradictions are capable of resolution. An *ive* generally seems to be preceded by a consonant which is affixed to it (res/tive, mo/tive, even pas/sive), but logical algorithms to differentiate the rules to handle apparent inconsistencies may become very involved. The plain fact is that the English language permits the division of words in singularly arbitrary ways and the rules are not always susceptible to analysis on the basis of the facts at hand since they have something to do with the pronunciation of particular vowels, the derivation of the word, or its part of speech. The proofreader might know the rule that *ing* stands by itself when it follows a double consonant that exists in the root word (pass/ing) but that there is a division between double

identical consonants when the root word does not contain both letters and the second is added as an aid to pronunciation (let/ting).

There are still other aberrations. There are differences between British word division and American (pro/gress and prog/ress for the noun; pro/cess and proc/ess for the verb). The British pay more attention to the derivation of the word (demo/cracy instead of democ/racy, know/ledge instead of knowl/edge). And even in the United States more emphasis upon word structure and less upon pronunciation is reflected in the word division principles of Webster's "Second" as compared with Webster's "Third" dictionary. In the latter, more and more popular as a guide in the United States, pronunciation tends to take over, so that the word becomes concor/dant instead of concord/ant, and depen/dence instead of depend/ence.

Difficult as it is to program good word division, at least it does not present the complications of Swedish, where the rule is that there are certain letters that must be repeated when a word is hypenated. And so *glasstrut* (ice cream) becomes *glass/strut* when it must be split. A similar condition occurs in German.

But our description of the problem is not yet over. The rules draw a distinction between syllabification and hyphenation. (The reader is warned that many dictionaries do not!) The Chicago Manual of Style specifies, for example, that word division should be made after a vowel "unless by so doing, the division is not according to pronunciation. Where a vowel alone forms a syllable in the middle of a word, run it into the first line."[1] Hence *commemo/rate* is better than *commemorate* and *physi/cal* is to be preferred to *phys/ical.* On the other hand, *ac/etate* is probably better than *ace/tate,* and *Bab/ylonia* might be preferred to *Baby/lonia,* although neither is a very happy choice.

1. University of Chicago *Manual of Style*, 12th ed., University of Chicago Press, page 137.

Where there are two vowels that are pronounced as separate syllables, they should be split, as *initi/ative rather than initia/tive.* Furthermore, compound words should be split, where possible, so that their elements are kept together. *Ante/cedent* is preferred to *an/tecedent* and *homo/geneous* to *ho/mogeneous.* Where words are joined by hyphens already, one should avoid placing another hyphen if at all possible (*quasi-public,* not *quasi-pub/lic*).

One wonders whether *any* algorithm can do justice to the problem of hyphenation. Moreover, there are the homographs, which generally hyphenate differently according to their meaning. A list of these, prepared by Howard Bezanson of Waverly Press, was once provided as follows:

> acerous, adept, agape, anchorite, arsenic, associate, bunter, coster, crater, decameter, denier, desert, divers, dosser, draper, elipses, ergotism, ergotize, evening, former, founder, fuller, gaffer, gainer, gaiter, genet, genial, gilder, graining, groper, grouser, halter, hinder, homer, hurter, impugnable, invalid, leaguer, legate, limber, luster, master, minute, nestling, petit, periodic, pinky, planer, prayer, precedent, present, project, produce, product, progress, putter, putting, rafter, rebel record, refuse, resume, rosier, spiller, stater, stingy, stover, tamper, tenter, and welter.

To this list of 70 we add *baton, Arundel, recreate, decanel* and *Eunice.* In addition, there are many proper names that may be confused with other nouns, such as *Weller, Baker, Miller* and *Wheeling.*

Obviously nothing can be done about these homographs except to flag them when they are divided, or to instruct the keyboarder to insert a discretionary hyphen in case they should fall at the end of a line. A contextual analysis would obviously be too complicated for a production typesetting program.

The exception word dictionary

In view of all the problems, it is not possible for a logical program to yield good hyphenation results, if our criteria for *good* results take *frequency, accuracy, throughput speed* and *memory requirements* into account. The safest thing to do about a combination of letters which break differently under different circumstances is not to hyphenate at all. If the program rejects all such conditions, the frequency of hyphenation will be extremely low. While too frequent word division is undesirable (such as three or more lines in a row), the occurrence of hyphens should be determined by parameters and not by the exercise of caution in a cowardly hyphenation routine. The program should be good enough to find most of the valid hyphenation points in almost every hyphenable word. If it fails to do so, poor justification will certainly rear its ugly head at occasional or even frequent places within the output.

Accuracy is important not only because a bad hyphenation detracts from the authentic nature of the product, but because to correct it requires the resetting of *at least two lines.*

Throughput speed and core requirements are factors which profoundly affect the cost of the typesetting system, and the former may be of importance where an improvement in turnaround time provides one of the justifications for the installation of a computerized typesetting system.

One reaches a point, therefore, where further improvements in the logical aspects of the program, undertaken in the interest of greater frequency and accuracy, make great demands upon throughput and memory, and it therefore becomes expedient to try to deal with the complications of hyphenation by using an exception word dictionary to supplement the logical program. Unfortunately, it is generally true that those who, in the interest of economy, cut down on the space devoted to the hyphenation algorithm, for the same reasons cut down on the size or availability of an exception word dictionary.

When such a dictionary is used, the word which straddles the fall-off point is usually first sought out in the stored dictionary. If it can be found there, the possible hyphenation points are given. If it cannot be found it is assumed that the word may be divided reasonably according to the logical rules of the program.

One of the earliest routines for hyphenation was developed by Perry Publications of West Palm Beach, Florida, in 1962, for an RCA 301 computer. A large dictionary was stored in an optimized fashion on a magnetic tape which could be searched by "read forward" and "read reverse" instructions. The words on this dictionary were those which could not be broken correctly after the third, fifth, seventh, or ninth character. *Sedentary* would not be found in such a dictionary, nor would *secretary* or *boreal*. Some words that did not follow the 3-5-7 routine might also not be considered exceptions if some other criteria kept the routine from doing any harm. For example, *cactus* divides after the third letter, and while it should not be divided after the fifth, the routine that requires two or three letters to be carried over would prevent a hyphen except after the third character.

Since there were many exceptions, this simple logical program needed a big dictionary. If the logic were more elaborate the dictionary could be smaller, but one unfortunate implication of an exception dictionary is that it almost always has to be searched before the logical rules can be implemented. Even if the algorithm is reliable enough to produce the desired result, the dictionary look-up procedure is usually not avoided in any event. Another limitation is the assumption that if the word is not specified as an exception, it follows the rule. On the other hand, the word simply may not have been included, in ignorance, because it had not been encountered before, or for lack of room.

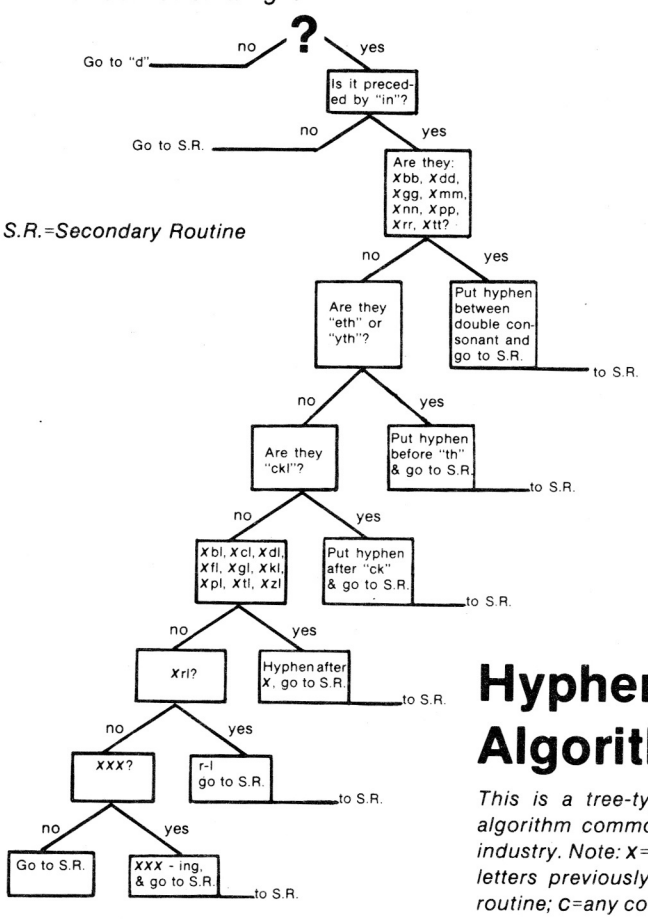
Exception words are usually stored in memory rather than on tape or disc because memory can be accessed more quickly, and can be indexed, if necessary for random access. Because of memory limitations not all exceptions can be stored, and newer programs are consequently making use of peripheral storage for this purpose. Whatever means may be used to store this dictionary, access time is exceedingly important, especially for those programs which run in an interactive mode. It may be less critical for programs which run in a background mode. Tape, which is inevitably sequential, is the slowest method of access. A fixed-head disc or drum arrangement would be ideal, but it is possible to make use of a conventional moving-head disc, especially if the program is "bright" enough to find something else to process while it is waiting the minute fraction of a second required to obtain the answer. In our own experience, a total dictionary which is able to locate any word stored on disc *with one disc seek* is faster than any combination of logic and exception word storage. The price one pays for such a dictionary is that it does take up a great deal of disc storage. Atex uses such an approach, and presently offers more than 100,000 words. If the average word consisted of ten characters, as much as one million characters of disc storage might be required. However, there are interesting methods of data compression available, such as not to use eight bits, or even six, to describe the 26 letters of the alphabet, and to use certain characters within the word for an address, so that they may be assumed and hence not stored.

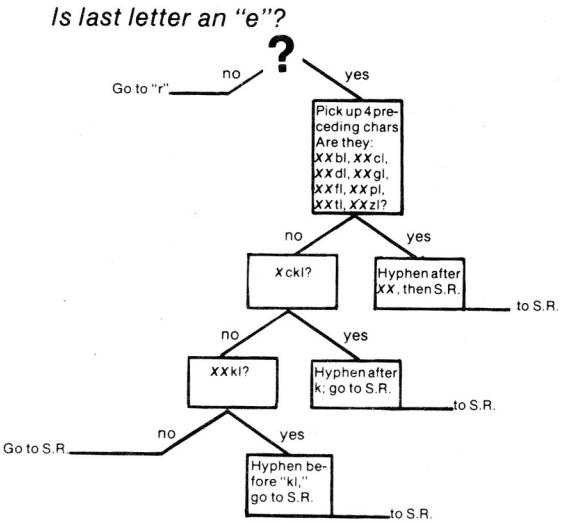
Generally, wherever a large dictionary is used and where very quick disc access is offered, the second price one pays is the time that may be required to

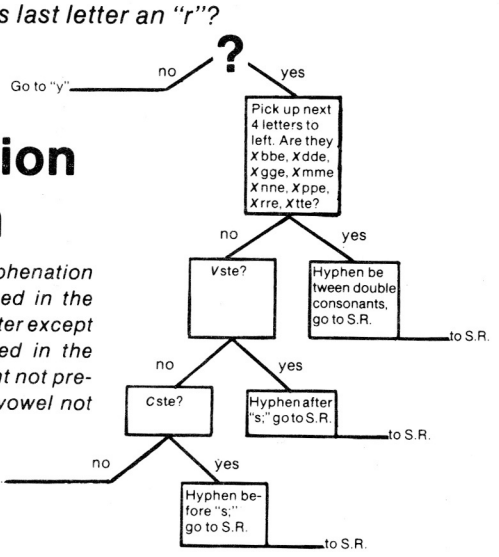
Hyphenation Algorithm

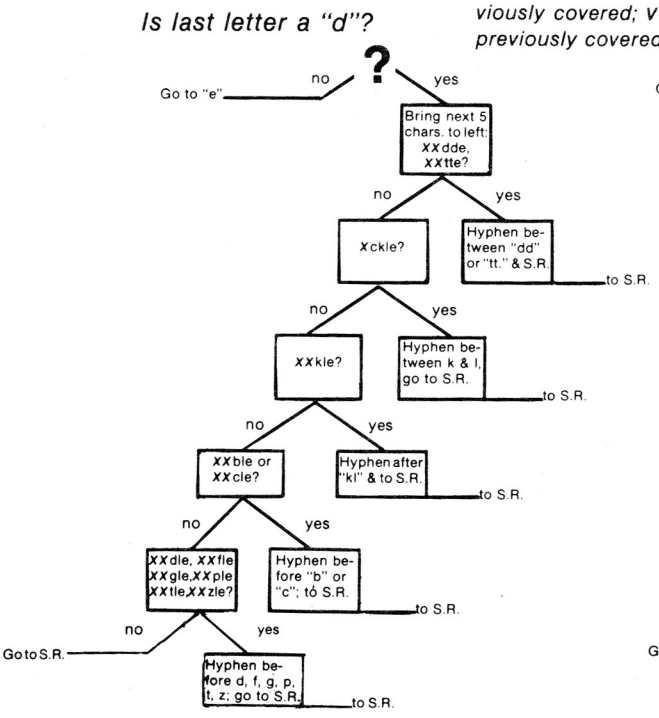
This is a tree-type hyphenation algorithm commonly used in the industry. Note: X=any letter except letters previously covered in the routine; C=any consonant not previously covered; V=any vowel not previously covered.

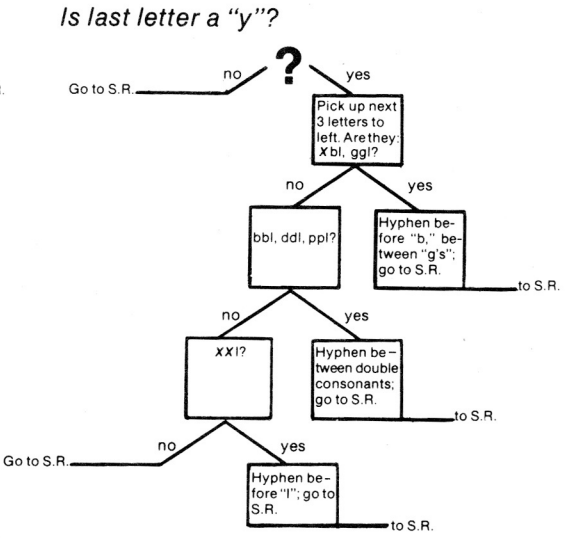
S.R.=Secondary Routine







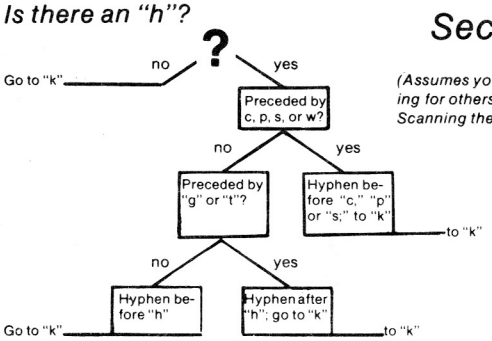




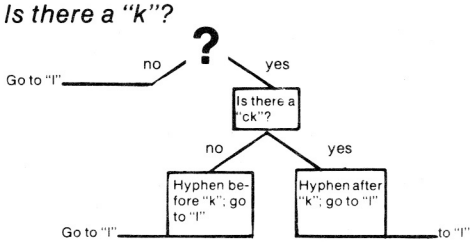
Secondary Routines

(Assumes you have already found a hyphen and are looking for others, or have not found one and are still looking. Scanning the word from left to right.)

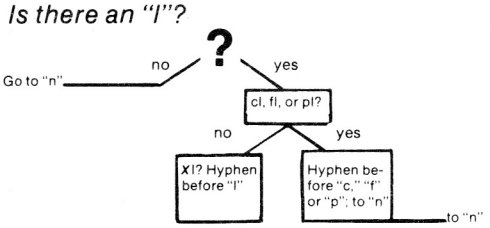
Is there an "h"?



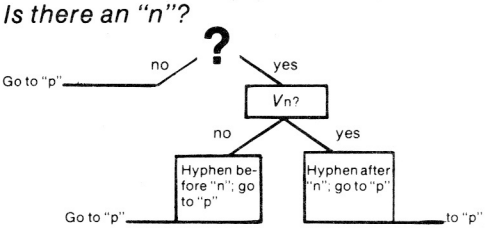
Is there a "k"?



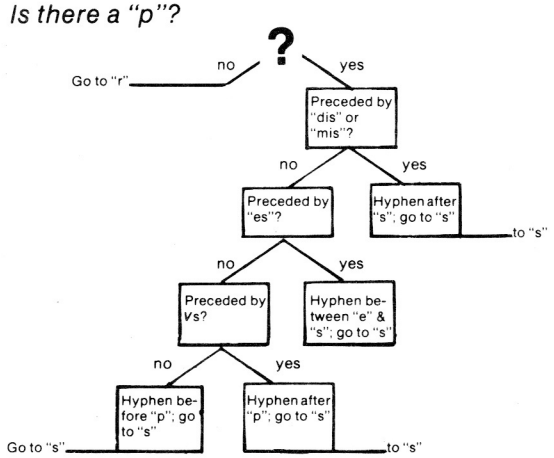
Is there an "l"?



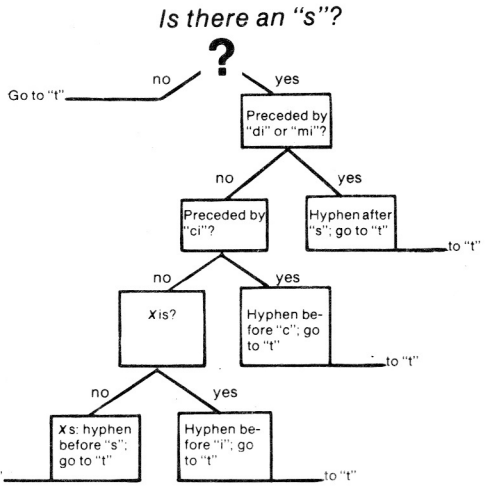
Is there an "n"?



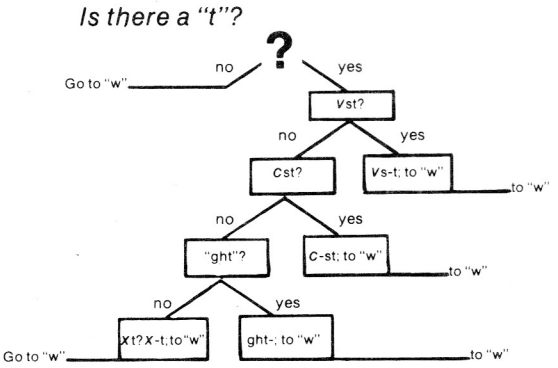
Is there a "p"?



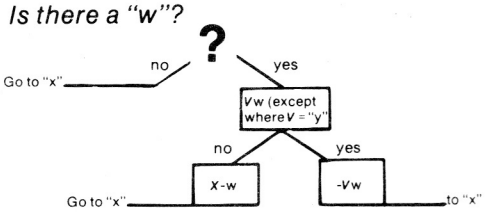
Is there an "s"?



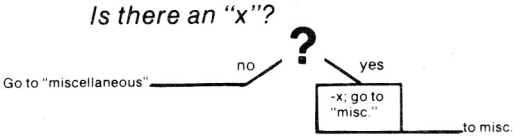
Is there a "t"?



Is there a "w"?



Is there an "x"?



Miscellaneous letter combinations within the word

Vb, Vc, Vd, Vf, Vg, Vm, Vn, Vs, Vz: Hyphen before *V*

Cb, Cc, Cd, Cf, Cg, Cm, Cn, Cs, Cz: Hyphen between the two consonants.

Note: *C*-r, *V*-r, *V*s-tr, *C*-str, ght-r, *X*-tr.

update the dictionary. This might necessitate that the entire system be appropriated for perhaps 30 or 45 minutes to run a "hasher" program against the master dictionary file.

The writer's strong preference is for the treatment of *all words* as exceptions, and even for the storage of preferred points for hyphenating the stored words. Various vendors, besides Atex, have moved in this direction. Some offer a 10,000- or 20,000-word "total" dictionary. Some offer a much larger dictionary. Those who seek to use this method soon discover that they simply cannot copy an existing dictionary. This is the case because most dictionaries do not, as a rule, provide true hyphenation points, but use syllabification instead. And very few dictionaries provide all the forms in which the word needs to be presented: computer, computers, computerize, computerizes, computerized, computerization, computerizations, perhaps even computerizing and computerizingly as well as decomputerize or noncomputerize and conceivably other compounds. Of course there would also be compute, computes, computed, computation and other similar forms. If one were assigning preferential values to hyphen points, one might store com(1)put(3)er and com(1)put(2)ers.

While many more suppliers will be moving toward larger and larger dictionaries, attention is still being paid to the development of more accurate hyphenation algorithms. The ones which seem most successful do not use word portions, but start at the back of the word (whether or not most of the word would fit within the line) and come to an understanding about the nature of the word by moving from left to right. If the last letter is an "s" the query is advanced, is the preceding letter an "s" as well? If so, are the two letters preceding these "ne" as in happiness? If not, is the third letter a vowel, *etc.* Working forward in this way, several vendors have managed to build up routines which take up less than 1,500 sixteen-bit memory locations and which produce moderately successful results—good enough, perhaps, for the average newspaper, and usable in a commercial environment if there is an easy way to search for hyphens and to correct bad ones by inserting discretionary hyphens, before the job is typeset.

We reproduce a set of logic rules in the accompanying diagrams (pages 230–231) to illustrate how such a hyphenation algorithm might be constructed.

Hyphenation verification

Another interesting approach, if a total dictionary is not used, is to cull out all of the words which have been divided, to keep track of their position in the text, and to print out the solutions arrived at. This not only serves as a check on the accuracy of the solution (and allows the user to locate the bad ones easily) but it can call attention to problems with the routine that may be capable of rectification. One such routine automatically checked all hyphens against a stored dictionary, and endeavored to move characters up or down within pre-justified lines, flagging those which could not be so adjusted. Usually a bad word break is only off by one, or at most two, characters.

Multi-line justification solutions

It is possible to have the best hyphenation routine in the world and still get results inferior to those achievable by a human, so long as the end-of-line decision is made in the vacuum of a single line of text. What makes the human operator's result better, in many cases, is his rejection of the line he has already composed. He then goes back and resets the *preceding* line, bringing a small word down or up, or dividing the fall-off word in a different place, to get better results in the line which follows. In the same fashion, we believe that a good computer routine should have the same capability, and should be able to "look at" five or six lines at the same time. If it can succeed in doing this, and if it used a total dictionary with preferred hyphenation points, no human would be able to obtain comparable results in a consistent fashion.

And so, as we have seen, the hyphenation of text is indeed a complicated and fascinating subject! For scholars and programmers there will always be scope for further research and experimentation.

Review Questions:

1. What criteria should be considered when setting up parameters for the allowable range of interword spaceband expansion for any given job?
2. What is meant by the term "straddle word"?
3. What are some of the elements that are involved in defining a word for the purpose of hyphenation?
4. Once a word has been defined, how does one go about the task of "cleaning it up"?
5. How many characters should be contained in a word before you consider it "hyphenable"?
6. At what point in a hyphenation routine would you perform a vowel group analysis?
7. What is the difference between a hyphenation point and syllabification point?
8. If your assignment were to devise an English language hyphenation algorithm which was only supposed to "look at" a limited number of characters in a word, would you work from left-to-right or right-to-left from the end of the word, or what portion of the word would you analyze? What criteria would you apply to find hyphenable locations?
9. What are some of the elements involved in a "word structure analysis" approach to a logic hyphenation routine?
10. What is meant by the term "homograph"?
11. If an exception word dictionary is used in conjunction with a set of logical rules, at what stage in the hyphenation decision-making process would the exception dictionary be consulted?
12. What is meant by a "total dictionary" approach? What are some of the problems associated with this solution to the hyphenation problem?
13. From your own favorite newspaper, bring in examples of bad hyphenation and see if you can figure out whether the errors were made by machines or humans.

15

Editing And Corrections

WE INDICATED in the preceding chapter that our discussion of editing and corrections would not be presented at one time and place in this book, but that its consideration would be scattered through most of the remaining chapters. The reason this is so is because the revision process takes place at various times and for various reasons. Let us begin by giving at least cursory consideration to some of the circumstances under which such activities may occur.

During the writing stage there will be many false starts, perhaps many drafts. Sentences, paragraphs and even entire chapters will be rewritten, discarded, rearranged, tidied, re-examined, set aside, resurrected, abridged, annotated, enlarged upon, clarified and mutilated. This is generally true even if only one author is involved. It is even more true when writing activity is carried on by a committee, or a group, or when the product has to be reviewed to conform to the requirements and expectations of participants in the process who bring widely different skills, enthusiasms and backgrounds to the collective undertaking. Writing is painful and tedious. It moves by fits and starts. At times it hardly progresses at all. At others it seems to retrogress. Then there are those inspired occasions when great forward strides are made.

Some writing activity is different from others. In general, narrative accounts such as will be found in the straightforward reporting of the news can be hammered out, at least indifferently well, by a skilled practitioner under the most adverse circumstances, by sheer force of will power and necessity. Wherever a formula exists for the accepted presentation of material—as is generally the case with the news—the writer at least has a frame of reference to ease his task. For other types of writing this is often not the case.

People not only have different styles of writing, but different habits. Some work from an outline. Others are unable to straight-jacket themselves with the bare bones of a skeletal structure, if we may indulge in a mixed metaphor. Some put a pen to paper. Others dictate to secretary or machine. Still others hunch

over a typewriter and, increasingly, nowadays, some record their thoughts and words—however tentative—directly into a computer by means of video terminals or "word processing" equipment. Eventually a rough draft will emerge. Some portions—perhaps much of the original draft—may be preserved. Or the entire effort may be discarded while the originator tries another tack.

All of this occurs during the writing stage. But who is to say what is writing and what is editing? The writer is his own best critic—or should be—and his own best editor. He determines what should be said and what should not be said. Later, subsequent editors may seek to improve upon his work. They may rearrange it. They may delete portions. They might conceivably enlarge upon or clarify an idea. But if a major overhauling of the manuscript is required, they will most likely return it to the author with suggestions or perhaps even directions.

The techniques we shall be discussing in many of the remaining chapters may be quite relevant to the writing process as well as to the so-called editorial process. They are, as we have suggested, indistinguishable, and editing never stops happening until the manuscript, or the galley, or the page, or even the printed product, is inexorably snatched away. During this entire awkward and anguished process some new tools now made possible by computer technology may be helpful or even essential.

For example, if new, clean versions of the author's draft can be provided to him, he may be able to follow his argument with new eyes, the better to evaluate his product. If what he has so far written can be faithfully reproduced free from error he will not be distracted or annoyed by what he or others may have done to his work. If, beyond this, when he makes subsequent changes they can be implemented immediately, and do not require interpretation or elaborate instructions which are conducive to misunderstandings, his activity is further eased and his productivity may be enhanced. If he can see his product in type, in galleys or in pages, he may be able to arrange his presentation to make it more forceful, since he can now imagine more clearly just how his work will be viewed by the ultimate reader.

But if the author does his own typing or re-typing of his manuscript, or even if it is done by his secretary, or a typing pool, or a free-lance typist, in preparation for submission to a selection editor or a copy editor, the very act of retyping will in itself generate new mistakes. Some of these can be caught immediately as they are being made. And so the correction process can begin at the time of initial input. If the capability exists to correct then and there that which is accidentally keyed—when the mistake is known to exist—obviously the subsequent editing or correction process is at least somewhat simplified.

Copy editing

In addition to the making of corrections during the writing process, and the implementation of self-corrections during the retyping process, there are copy-

editing or redaction corrections to be made. Preliminarily, these are made by the writer himself, but under the auspices of most publishing organizations, a copy-editing process takes place to assure that the writer has spelled and punctuated correctly, that his grammar, if not impeccable, is acceptable and comprehendible (a copy editor might suggest that the word should be comprehensible!) and that his choice of words is neither too monotonous nor too precious. Most of us have bad writing habits, or we are simply too lazy to try to think of ways of getting around the use of the same terms or catch phrases over and over again. (In writing *The Seybold Report,* which provides a semi-monthly analysis and update of computer typesetting and editing systems, we struggle constantly to find—and generally fail to discover—another word for "system.")

And so copy editing generates another cycle of corrections, and unless the copy editor can effect them himself (on-line to the "system") still another cycle of corrections may be required to put right the errors introduced during the correction process.

Copy fitting

Yet another level of corrections has to do with those changes necessary for purposes of *copy fitting.* Adjectives may be inserted or deleted to avoid "hangers" or "widows." Text may be modified, if permissible, so that two lines of a chapter ending do not carry over to a page all by themselves. Text which may not be modified may have to be reset tighter or looser, or made up differently in pages, in order to avoid such a condition. References must be added to adjoining illustrative matter, and footnotes must be located consistently within the page frame. Moreover, subheads, inserted into the text to break up the running matter and to emphasize significant subdivisions thereof, may have to be deleted and others added when they fall awkwardly at the bottom of a page or column. Finally, when all of these things have been accomplished, there can yet be another cycle if the author is given a further opportunity to see his product before publication.

This obvious summary of the correction and editing problem has been given to illustrate the point that, generally speaking, editing and making corrections are processes which do not generally happen just once. They are continuing activities and one editing or correction cycle may well grow out of another. (A mishyphenation which has to be corrected may cause a "hanger" to appear.) And so a composition "system"—especially one that conceives of the composition process as being part of a flow of copy from the writer to the printing press and thence to the reader—must contemplate the need to cope with this problem not just once, but many times—not only in the raw manuscript but in the final manuscript, not just in the manuscript but in the composition input, not merely in the input but in the output as well. Hence, an ideal solution—aside from one that minimizes the need for certain types of corrections—is one that makes possible the reprocessing of the most recent version in order to reduce the possibility of adding still new errors, and to abridge the recycling time as much as possible.

Corrections in photocomposition

Corrections have always been difficult to make in the photocomposition process because the compositor or keyboarder has not been able to see what he is doing. Only when his photographic film or paper has been processed is he able to get the "answer" to questions which may have arisen during the input process. (Did I strike the correct key? Is there in fact a bullet in this font? Did I remember to go out of bold and back into roman? Did I cancel the indent? Did I consistently describe the depth of the desired indent when I changed point size? Did I remember to align my numerals in that outline section?)

For all of the advantages of photocomposition—flexibility and speed, as well as quality and versatility—the "blindness" of the process, and often its roundaboutness as well—seem to magnify the correction and hence the editing problem.

Input perfection

It may be argued that the way to avoid the need for any corrections is to take pains to have "perfect input." Certainly, clean and perfect input to the photocomposition system will reduce the number of errors, but it will not eliminate the need for editorial changes (as for copy fitting purposes), nor will it resolve problems created by the composition process itself. Of course, it can be argued that perfect input would include not only the input of the manuscript itself, but also the input of all of the parameters that seek to cope with problems of composition. A fully-automated system can in fact do much of the required copy fitting, for example, if the problems of the job are adequately and correctly anticipated and described. We shall discuss this possibility in our chapter on Pagination Solutions.

Even before photocomposition was in wide usage, there were at least three methods of input perfection which were invoked. One is still commonly applied today. When punched cards provide input to a computer "system" it is possible to proofread the cards, and to remove one or more cards from the "deck," substituting others which have been totally re-punched. Consequently, one error will necessitate the rekeying of (generally) 80 keystrokes, but that card can then take the place of the one which contained an error, just as a slug of hot metal type could be removed and another substituted. The reading of punched cards is common practice in the data processing industry, and for even more accurate input a key punch *verifier* was developed several decades ago. One operator punches the data onto a card. This is passed, along with the initial manuscript, to another operator who works on a verifying punch. That person inserts the card which was first punched, at one location within the machine, and then inserts a fresh card at another location. The operator then rekeys and the device makes a comparison on a keystroke-for-keystroke basis. If the original card is incorrect and the operator strikes the proper key, the keyboard locks and data cannot be entered until a determination is made as to which data element is the right one. Thus the card input which emerges from the verifying punch is bound to be correct in all cases excepting the possibility that both operators might make identical errors.

Punched cards are sometimes used for running text, but they are more commonly used as input for "fielded" data information, such as a last name in one location, the given name in a second, the street address in the third group of allocated positions, *etc.* But cards could be used for text input if upper- and lower-case conventions are adopted. Cards are not a congenial method of text input largely because the character repertoire they can convey is so limited.

But a paper tape merge device was also developed. The Friden company, later a subsidiary of Singer, offered in the 1950s a means of perforating paper tape while providing a "hard copy" representation on the same typewriter-like keyboard. Thus it became possible to proofread the typewritten copy and to ascertain whether or not the paper tape contained errors. If such errors were detected, the original tape could be placed in a reperforating unit and played out a second time, producing a new typed copy, and punching out a new tape. The person who monitored the unit could watch the typing process as the paper tape activated the machine, stop the machine, correct the error which was about to be reproduced, by keying the word properly, and then cause the original tape reading process to continue down to the point where the next error appeared, thus creating a "clean tape." Obviously this process was slow and tedious. It also suffered from the inconvenience that there were certain input codes which had no hard copy or printout equivalent (such as an em space). Nevertheless, this technique was used in two kinds of situations: to improve or perfect input tape for a computer typesetting system, and to enter editorial revisions where rewriting took place, as for copyfitting purposes. In the latter case, for example, *Time* magazine, *U.S. News,* and *Newsweek*, to mention three publications dealing with more or less the same kind of copy-fitting problem, had their operators not only punch out a paper tape and provide them with a typewritten printout for proofreading and editorial purposes, but also had them prepare that tape on counting keyboards which required that the operator make end-of-line decisions. In this fashion, the editors could count justified lines to obtain the proper fit for the story, down to the very last character. The tape could then be played back on a reperforator which was also a counting keyboard, new words could be added, old words deleted, and new end-of-line decisions reached.

A somewhat similar but in some respects more efficient solution was a different kind of paper tape merge device which could be instructed to read from tape 1 up to a certain point, then to read from tape 2 for several lines, and then to return to tape 1. Two tapes—the original, and the lines or words to be substituted —were thus converted into one "clean" tape.

The computer as a "tape-merging" device

In the early days of computer typesetting, especially during the period from 1963 to 1970, the computer which performed the hyphenation and justification processing was also used (in some cases) to merge corrections into the text and code-command stream. Several techniques were employed:

1. The original paper tape was read by the computer paper tape reader, and a computer printout was produced by the on-line high-speed printer. The paper tape was saved and identified. The computer printout was proofread. Generally, the computer printout broke the text into numbered lines. Usually these lines had not yet been hyphenated and justified. Generally, the on-line printer which produced the line-numbered "proof" output broke the line after about 80 characters, but at the end of a word—that is, on the interword space nearest to 80 characters (or some other arbitrary number approximating a workable line length). This "printout" would then be proofread and marked with corrections. A new paper tape would then be punched containing correction messages. These consisted of the number of the line to be changed and the correction to be made. In some cases an entire line was substituted. More commonly, the word to be corrected was identified by position or by unique character string, and the correction to be substituted was then indicated. The format of the correction message was generally either:

   203; teh, the—where "the" was to be substituted for "teh"; or
   203; 0304, the—where "the" was to be substituted for whatever appeared between the third and fourth interword spaces in line 203.

   The correction tape was then read in and stored in computer memory. Then the original tape was read, and the corrections were identified and inserted into an allocated area within the memory of the computer as the "old" material was examined, line by line, and passed on through the system. Since the memory was of insufficient size to store the entire text file, a new, "clean" tape was punched out, and that tape served as input to the h&j routine.

2. Another variation was to equip the computer with two paper tape readers and one paper tape punch. The first entry on the correction tape would be read to ascertain the location of the first correction. The original tape would be read in and punched out up to that point, then the correction would be inserted. This method had the advantage of not using up so much computer memory in order to store correction messages. However, in both cases the corrections had to be entered in serial fashion, and if the lines to be corrected were misnumbered, either the correction would be inserted at the wrong location, or other succeeding corrections would be rejected. In other words, if you had prepared correction messages for lines 24, 28 and 30, but addressed them as 34, 28 and 30, the correction for line 24 might be applied to line 34, and the messages for lines 28 and 30 would be rejected as out of sequence.

3. Other, more sophisticated, correction techniques were also employed. For example, the initial input tape, when read in to provide a computer printout for proofing, could also be written out to magnetic tape, or stored on disc. Then when the correction run was to take place the paper tape containing the corrections would be read by the program along with the stored text from tape or disc, and a new version would be prepared and stored on another magnetic tape, or at another disc location. Since magnetic tape can be

```
00001  +#T&HIS IS A SIMULATION OF AN ALL-UPPERCASE PRINTOUT SUCH AS WAS FREQUENTLY PROOFREAD

00002  DURING THE EARLIER DAYS OF COMPUTER COMPOSITION. #H&OWEVER, THE REPRESENTATION OF UPPER

00003  AND LOWER CASE CONDITIONS IS PURELY ARRBITRARY.  #T&HIS LINE CONTAINS AN ERROR.=

00003,06,07 arbitrary.===

00001  +#T&HIS IS A SIMULATION OF AN ALL-UPPERCASE PRINTOUT SUCH AS WAS FREQUENTLY PROOFREAD

00002  DURING THE EARLIER DAYS OF COMPUTER COMPOSITION. #H&OWEVER, THE REPRESENTATION OF UPPER
                                          *********
00003  AND LOWER CASE CONDITIONS IS PURELY ARBITRARY. #T&HIS LINE CONTAINS AN ERROR.=
```

Note: The first example shows the original printout. Then a correction message has been keyed on the input perforator, addressing the line number and the location of the word, and substituting the desired word. Finally, a corrected printout is provided overprinting with asterisks the word that was corrected. It would be possible to suppress printing all lines that did not need to be corrected, and also to print out correction messages that were rejected as being out of sequence or faulty in format.

```
00001  This is a simulation of an upper and lowercase printout such as was proofread during

00002  a later period in the computer composition evolution. Upper case differentiation is

00003  now resolved but symbols over or under the letters may be shown to indicate italic or
                                                                                  111111
00004  bold or bold italic characters, and special symbols may also be necessary to indicate
       2222    33333333333
00005  characters that require special treatment, such as for the accents on the word eleve.=
```

read at higher speeds than paper tape, and the information is recorded more accurately, and since output to magnetic tape can be written at substantially higher speeds than paper tape can be punched, the use of magnetic tapes or disc storage substantially speeded up the correction process.

Such "correction runs" were of course performed in a "batch" environment —that is, the entire job had to be processed serially from beginning to end. In some instances there were programs which permitted these runs to be carried forward in a "multi-tasking" environment—which meant that some other kinds of programs might be run simultaneously (in another computer memory partition). For the most part, however, in the earlier days of computer typesetting, the computer was "dedicated" to the performance of these correction runs while they were mounted, and it was totally, but inefficiently, occupied in the performance of this task.

Corrections to the output of h&j

Most of the computerized correction merge programs which were used in the late 1960s and the early 1970s operated on the "file" before it had been processed through hypenation and justification and could not update the h&j'd file subsequently.[1] This is because the file structure was usually substantially changed during h&j, and perhaps even the internal coding conventions were different. Unfortunately, however, many errors in the file could not be detected until h&j had been performed or attempted. Since composition coding commands are quite complex, and were often input or even conceived incorrectly, the h&j might

1. This was not true of the correction routine developed by the firm founded by this writer in 1963. The Rocappi correction routine, which operated in a mag tape-to-mag tape mode, and which was the industry's first "correction merge" program, could in fact correct and reprocess the output of the h&j'd file.

"hang" and require computer operator diagnosis to determine what went wrong. (The input specified a two em indent in a 10 *point* measure, for example.) Or the program would be aborted with or without diagnostic identification of the nature of the error. In either case, one would have to go back to the input file, find the mistake, correct it, and re-run the job through h&j. But if the errors subsequently detected did not interfere with the initial h&j processing, it obviously would save time and effort if the most recent version could be revised. (There was a paragraph set in bold which should have been in italic, for example.) It would have been much more convenient to correct the output of h&j, but very few programs were sophisticated enough to permit this.

Some practitioners found that proofreading was performed more accurately if typeset output was read rather than a computer printout. Of course, the computer printouts improved as time went on. Upper- and lower-case character repertoires were developed. Overprints and underprints identified changes in type faces. Statistics were sometimes generated to signal whether the justified lines were so loose that they required letterspacing. Nevertheless, typeset output is certainly easier to read, and some programs offered a proof version to be set in galleys, in the actual type of the job, but with line numbers to permit ready identification of locations to be corrected. When this was done, depending upon the computer typesetting "system," one of four alternatives was available:

1. You might have to go back to the initial input and correct that. This could mean searching through a line numbered file which did not agree with the final output since h&j obviously caused the lines to break differently. This was indeed an awkward and time-consuming solution.

2. The system might generate two outputs: one in a "language" to drive the typesetter, and a second version being created to correspond with the h&j'd output, and broken line-for-line in the same manner, but in a version which remained in input coding and input file structure conventions.[1] It would be this alternate version of the file (sometimes called the "pseudo-input" although "pseudo-output" would have been a better term) which could be corrected.

3. A few systems provided for h&j'd output which was reprocessable. The version stored (on paper tape, magnetic tape or disc) and which drove the typesetter, could be fed back into the system for corrections and re-h&j-ing despite the fact that it contained output typesetter driver codes. These were stripped from the file when it was read back in.

4. One or two systems separated the h&j pass into two portions: a generalized h&j output could be created, which was in fact reprocessable, and a subsequent output driver program could convert this file into the language required for a particular typesetter. Consequently, it would be the "interim" file which would be reprocessed, but it was this file which was always saved. And being device-independent, it could even be processed to drive a different typesetter. The output file, which was device-dependent, was "thrown away" after it was used.

1. Alternatively, the file could be kept in an internal language coding structure which differed from both input and output, on the assumption that input coding would be translated into the "main line" program language in any event.

While we have described all of these alternatives in the past tense, let it be said that these same choices are generally available today. Some systems use one technique and some another.

But if editorial changes are to be introduced for copy-fitting purposes, then correction programs are required which offer reprocessability, and which do not make it necessary for the user to revert to the initial input file. Some of the more sophisticated systems we shall be describing subsequently also offer *partial processability*—that is, the ability to define only that element within the text which requires a change, and to by-pass all other portions of the job. This is a more difficult and hazardous undertaking than might appear at first blush. Text processing is inherently a serial operation, because a code which is once embedded (in effect) sets a switch which governs until further notice. That switch might call for a change of point size, or measure, or type font, or might introduce an indent, or suspend hyphenation, or go into ragged setting. Unless the entire file is re-examined, the selection of a single paragraph to reprocess, out of context, may be predicated upon assumptions as to format conditions which are not in fact valid. (This risk can be avoided if all conditions are periodically restated, or are carried at all times with each line of the text file.)

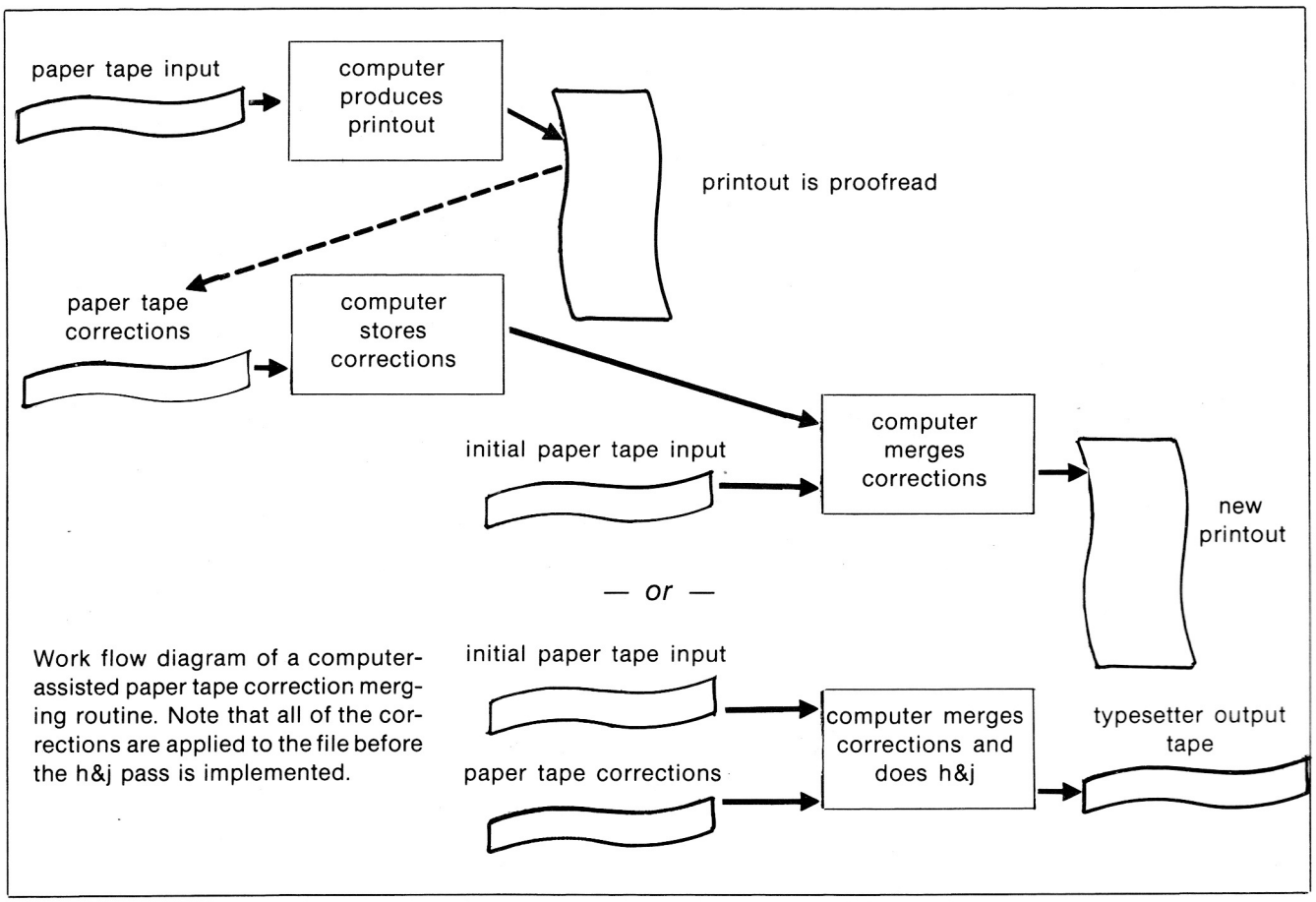
Limitations of correction systems

All of the correction procedures we have discussed so far have serious limitations. Let us attempt to categorize them and indicate the problems associated with them:

1. *Pre-processing "tape perfection" schemes.*—These are solutions that involve making corrections to the input tape prior to feeding it into the computer, or going back again to the input tape after an unsuccessful or disappointing processing run, finding the mistakes and changing them. These methods could embrace the substitution of one punched card for another, of printing out and punching a new paper tape containing unchanged and altered material, or of merging two tapes. Limitations are:

    a. The corrections must be serial in nature, from front to back of the job.
    b. The corrections must be located by visual inspection of an unjustified printout or hard copy.
    c. It is impossible to anticipate whether the corrections will solve the problem, particularly if the problem relates to copy fitting—*i.e.*, the impact of subsequent h&j upon the input.

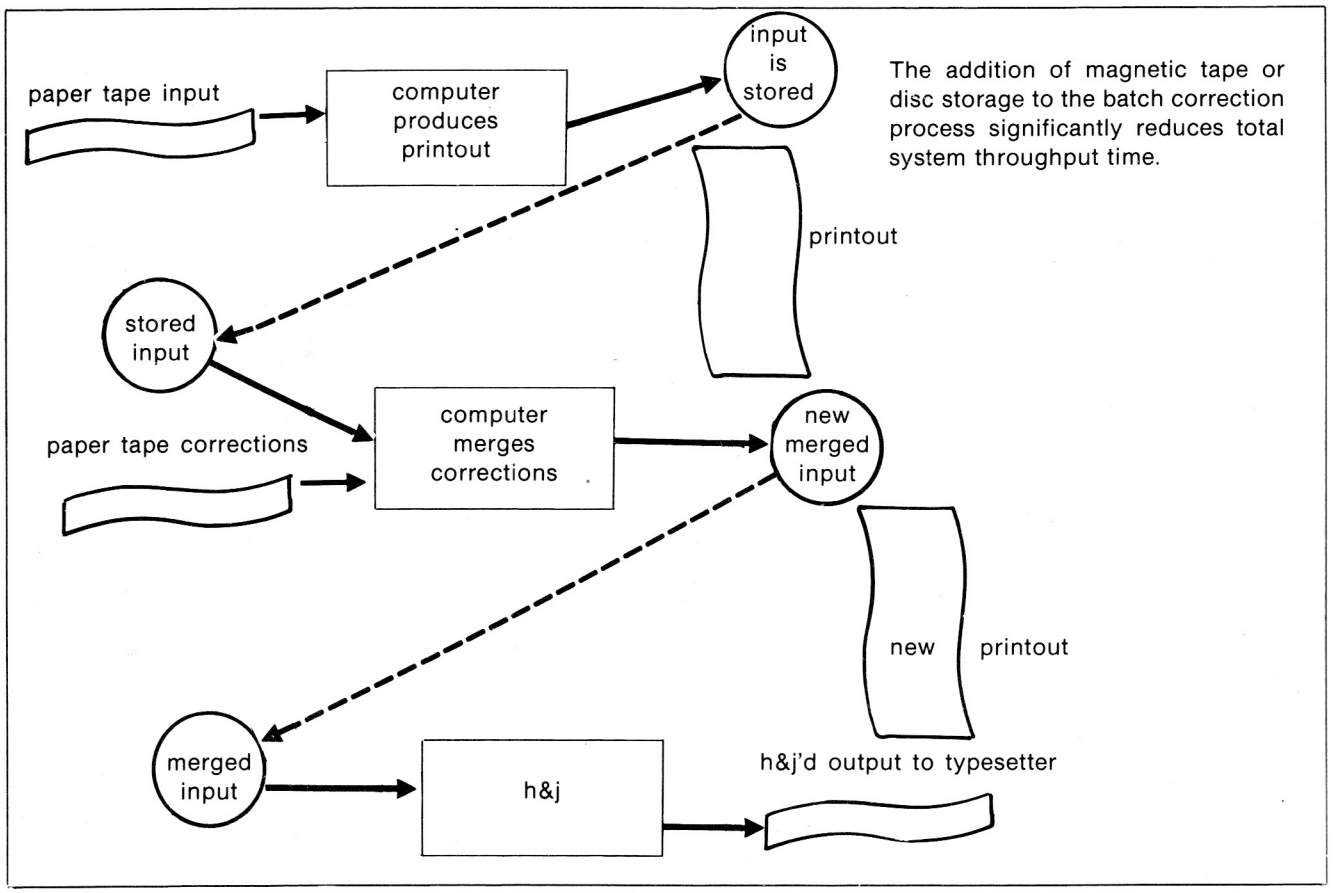
2. *Pre-h&j computer correction merging schemes.*—Such schemes use the computer to accomplish more quickly the kind of serial correction techniques described above. Aside from the fact that this "batch" correction technique is faster than those listed immediately above, it is subject to the same limitations.



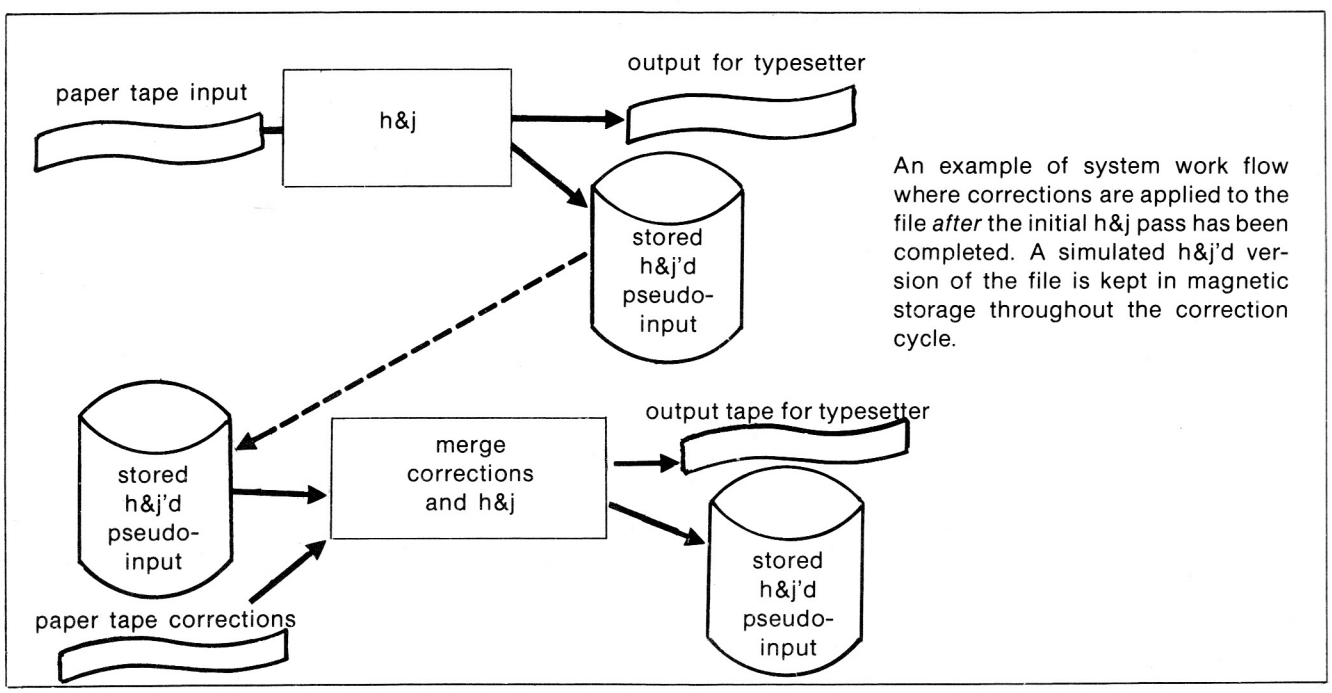
Work flow diagram of a computer-assisted paper tape correction merging routine. Note that all of the corrections are applied to the file before the h&j pass is implemented.

3. *Post-h&j correction merging schemes.*—Mistakes uncovered by h&j, or changes which become apparent after h&j has been run (as for copy fitting) can be corrected by going back to the initial input, or a "tape-perfected" version thereof, but it is more expeditious, and generally easier, to correct a post-h&j version. As we have indicated, this may be a "pseudo-input" version, an output driver-compatible version, or a file that stands somewhere in between. Limitations are:

    a. The process is still generally serial in nature.
    b. One must still speculate what the output of the new h&j pass will be.
    c. It is a batch process. When it has been completed the results can be studied. If they are still not satisfactory a second pass could be run, on the new h&j'd output, or on the previous version if it still exists and if old and new changes are combined. In this connection we must note that there are some programs which permit the making of corrections to the correction input, rather than to the file itself. In other words, the proposed file of correction messages can be proofread and corrected (often by the same program) before it is applied against the text file.



The addition of magnetic tape or disc storage to the batch correction process significantly reduces total system throughput time.



An example of system work flow where corrections are applied to the file *after* the initial h&j pass has been completed. A simulated h&j'd version of the file is kept in magnetic storage throughout the correction cycle.

Relationship to page makeup

If a page make-up program exists, it would be advantageous to apply corrections to material which had already been processed through hyphenation and justification and also paged, if the corrections would not affect the pagination, or through hyphenation and justification and repaging, if they would have an impact upon the arrangement of text within the pages. To go back to initial "input" would introduce uncertainties and involve extended processing cycles. How corrections can be introduced in paged material will be covered subsequently in our discussion of pagination.

Word processing and time-shared text processing programs

We must, however, accept the fundamental principle that photocomposition makes the greatest sense when two conditions can be met: input can be corrected so that output is clean, and output can be produced in fully made-up pages. To accomplish these objectives, at least some correction or editing programs must be an integral part of a composition "system." This would not preclude the desirability and use of other kinds of editing and correction techniques. These would be "front-end" systems—that is, systems or methods which would help to provide input to the composition system as clean and perfect as possible. Office word processing equipment can be used in this fashion, and the material so stored may be made compatible with the typesetting system so that rekeyboarding can be avoided. Another alternative would be to input on line to a computer and to call the text back from tape to disc for subsequent corrections and editorial revisions, prior to the typesetting format runs. Time sharing terminals may be installed in your own home or office for this purpose.[1] But manuscripts which are supplied to the computer typesetting process in machine-readable form by one of these methods (*e.g.*, word processing or time-sharing) may not contain the necessary typographic encoding. If they do not, it will be necessary for the composition facility to provide a further editing capability in order to insert these highly important flags and instructions.

1. Time-sharing networks frequently offer such text-editing programs as ATMS (Advanced Text Management System) or WYLBUR for remote text entry and editing.

Manual stripping vs. new, clean output

Despite our assertions above, it may be easier to re-set only those lines or paragraphs that require changes than to go back through the processing cycle in order to merge in corrections and come out with new, clean galleys or pages. Even where correction programs exist and are used for a great deal of the processing activity, stripping corrections will surely be made under some circumstances. However, if the intent is to maintain a clean "file" for future use, the stripping solution will not serve the purpose. Until the advent of the video terminal, however, the stripping in of corrections on "one-time"—as opposed to "data base"—typesetting was common. The video terminal has changed all of this. Its impact upon the industry—from the editorial stages through production—has been profound. But before considering this device, and its impact, we shall have to digress to provide more information about the input process, which we have so far taken pretty much for granted.

Review Questions:

1. Delineate the various levels or cycles of corrections which might occur as a manuscript progresses from original conception to final publication.
2. Describe some of the means originally used for "input perfection" of material keyed on punched cards or paper tape.
3. Why is the term "line editing" used to refer to the operations performed in computer-assisted tape-merge correction routines?
4. What advantages are to be gained by applying corrections to the "output of h&j"?
5. If the h&j'd output version of the file is to form the basis for future changes and corrections, how can the output driver coding be handled so that it does not interfere with the proofreading or the reprocessing of the file?
6. What are some of the file handling implications involved if "partial processability" is desired?
7. Describe the inherent limitations for each of the correction procedures discussed in this chapter.

16

Input into a Typesetting System

EARLY COMPUTER TYPESETTING programs often assumed that input would correspond with output so far as coding and character representation are concerned. For example, the program and system might take for granted that the input would be generated by TTS-keyboarded paper tape, with the same codes and methods we described in Chapter Six, and the output typesetters would likewise be driven by the same six-level TTS paper tape. The primary difference between input and output would be that the output would contain end-of-line decisions inserted by the computer, and perhaps some other formatting conventions, such as indents which would involve the automatic generation of a fixed amount of white space at the beginning (or end) of a line.

It is true that most of the early typesetters—even the phototypesetting devices—were TTS compatible, but it quickly became apparent that the composition system would do well not to assume a certain input coding structure, just as the output of the system should not assume a specific output driver coding structure. Consequently, typesetting systems became somewhat more modularly designed, translating one or more input coding sets into an internal processing language, and translating again to an output driver coding set.

But it is not sufficient for the interfaces between the input module and the main line program, on the one hand, and the main line program and the output module, on the other, to perform a code-for-code translation, as between TTS and ASCII. It is not even sufficient to perform a conversion, say, from six-level to eight-level coding, which might involve differentiating upper-case characters not by a precedence code (the "upshift") but by changing the bit pattern which distinguishes an "a" from an "A"—as by adding one higher-order bit to the same basic lower-level code. Strings of codes may be converted into a single code "going in" and a single code may be converted to a code string "going out"—or *vice versa*.

The capability of translating not only one code to another, but one string of codes to something else, offers great flexibility to the solution of input problems,

and permits a variety of different approaches to be employed, even by the same user, in order to meet different needs.

For every application there is, presumably, a best solution, but it will have to take into account factors of human convenience and habit as well as ideal theoretical conceptions. Although most coding conventions today are expressed in an eight-level "byte," a keyboard with 256 keys, or even 128, may not be a good idea. If the keyboard layout bore no resemblance to any other keyboard, and its use would require substantial retraining of writers and operators, that keyboard would certainly have to offer a demonstrably better solution to be worth the retraining effort.

On the other hand, it may, in fact, be easier to use a keyboard with only ten keys, all of which rest immediately under the fingers (something on the order of a Stenotype), and to strike combinations of these ten keys, rather than to search for the proper key on a larger keyboard.

A good solution will be oriented toward the user's specific applications, and yet there should probably be some degree of commonality between all approaches: they need to be reducible conceptually to some basic structure which will facilitate data interchange. In order to achieve this objective we will need to reach some kind of agreement on conventions for typesetting and related information interchange. Conventions, we are reminded, need not be binding. Yet there are times when it is essential to be certain that we are understood. It has been suggested that a proper young lady in the United States would hardly leave word with the hotel room clerk that she would like to be knocked up at eight o'clock in the morning! Such a request in Britain might be disappointingly commonplace, and not because of a difference in moral standards. It is the use of the conversational (information interchange) idiom that causes the confusion. Our own mental computers happily make allowances for some of these contemporary idiosyncrasies, invoking a translate routine which can cope with phrases and words as well as individual graphic symbols. If we wish to write for posterity, however, we had better know which of our current "in" expressions future generations will be able to "dig" and which ones will "bomb out."

Our need for conventions relates to two primary areas of information interchange. One has to do with the presentation of the character repertoire itself. The other is concerned with a definition and identification of typographic "situations."

Let us look first at the problem of the character repertoire. If we can agree to group certain characters into families or "sets" we can greatly simplify the problem of character access. For example, there are 44 common keys on a typewriter, plus a space bar and certain "function" keys—backspace, shift, tab set, tab clear, tab and margin release or reset. If our typewriter is a Selectric, with a "golf ball" element, we can design our own golf ball, and access any characters we want on that new golf ball if we know what key it is we need to strike. We might have a Greek element, and another for mathematics, and still another for foreign ac-

$$A\ B\ \square\ \Delta\ E\ \Phi\ \Gamma\ H\ I\ \Xi\ K\ \Lambda\ M\ N\ O\ \Pi\ \Theta\ P\ \Sigma\ T\ \Upsilon\ \Psi\ \Omega\ X\ \llcorner\ Z$$

$$\alpha\ \beta\ \varphi\ \delta\ \epsilon\ \phi\ \gamma\ \eta\ \iota\ \xi\ \kappa\ \lambda\ \mu\ \nu\ o\ \pi\ \theta\ \rho\ \sigma\ \tau\ \upsilon\ \psi\ \omega\ \chi\ \vartheta\ \zeta$$

$$\rightarrow\ \infty\ \triangle\ \|\ \nabla\ \not\propto\ \propto\ \propto\ \perp\ \not\perp\ \doteq\ \simeq\ \angle\ \zeta\ \pm\ \mp\ \Omega$$

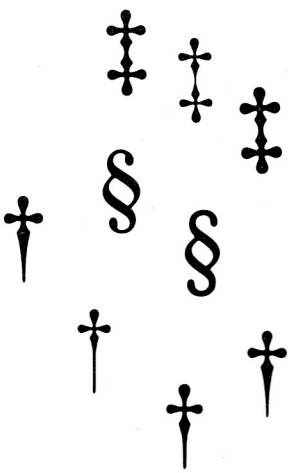
$$\int\ \uparrow\ \downarrow\ \leftrightarrow\ \circ\ f\ \searrow\ \nearrow\ \surd\ \times\ |\ |\ |\ \not\triangleright\ \not\triangleleft\ \geqslant\ \leqslant\ ::\ \because\ \therefore\ :$$

$$\div\ \gg\ \ll\ >\ <\ g\ \Re\ \pounds\ /\ .\ \partial\ —\ -\ -\ _\ \cdot\ +\ \oint$$

It is often necessary or desirable to access a large character repertoire from a single keyboard. Greek characters, logic notation, and mathematical symbols constitute just a few of the many possible character sets that might be available.

cents, and perhaps one with all kinds of interesting special characters we might need for scientific or logical representation. Suppose we worked out ten different golf ball elements. Apart from our preferences for type faces and type styles, we could probably cover most of our typographic needs merely by mounting such new typing elements. We would soon learn which character set was which, and which keys to strike to access the desired symbols. But if everyone designed his own character sets, with entirely different groupings of characters, each positioned differently, moreover, on the typing element, it would be tedious indeed to locate the characters we wanted to use. We would know neither which golf ball to mount nor which key to strike.

However, a computer can cope with this problem better than a mere human. For every user it would be necessary to input an elaborate translate table, especially made up for that one application, listing every one of perhaps 880 different characters, and identifying the particular golf ball and the particular position on that element for each such character. Since the character repertoire is open-ended, the translate table routine would also have to have an infinite capacity for expansion. How much better it would be to agree on at least a half dozen families or groupings of characters that would mean the same for everybody, even if they did not accommodate each individual's needs quite as readily as his own special-purpose solution!

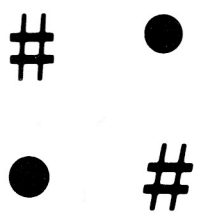


What is true on the "input" end may also be true on the "output" end. Each user may have a need for a slightly different character set, or, at least, one arrangement may be much more efficient for his requirements than another. If he is setting stock market quotations he would want to make extensive use of fractions, with special symbols such as bullets and daggers. If he is setting legal statutes he might prefer to give up a convenient location for fractions in favor of having the section mark, the paragraph symbol, and call-out references for footnotes. So what is important is not merely which characters are available, but how conveniently they may be accessed. This is true both with respect to input, and also for output.

Input keyboards can indeed be tailor-made, and so can output typesetter font arrangements. But customization is expensive. Yet a case can be advanced for what we may call "position keyboarding." By this we mean that we can visualize a keyboard as consisting of a certain number of addresses. We might simply number each key, beginning at the upper left hand corner, and let that number correspond with a position on the output device's font strip. Hence we would know that if we struck key number 22 we would always access character number 22 on the output device (and if we had an interim proofing arrangement, such as on-line printer, we would also want character number 22 to correspond at this level as well).

An alternative to position keyboarding would be to expect that the computer look-up tables which stood between input and output would make it their business to know what characters we intended, and to find them on the output device. We could therefore devise a "pseudo-font" for input and internal processing, and then, on the output pass, "map" the characters we wanted to whatever location they might occupy on the film strips of the output typesetter. In fact, we could economize in the disposition of our graphic images. Conceivably, for all fonts and character arrangements we would require only one period, except perhaps to differentiate bold from roman type. On the input we would not have to shift from one keyboard arrangement to another to access that period. (We would have a period on every keyboard layout). But so far as output is concerned, the program would find the period's location and put out the appropriate code or string of codes to access that character.

This concept is not as far-fetched as might at first appear. There is an increasing tendency to arrange character and keyboard layouts so that the operator no longer has to worry whether he has a "bullet" in a particular font, or whether he will have to transfer from font 1 to font 3 and strike the upshift of the "3"—which normally contains a number sign (#)—to access a bullet. With only one bullet in his type library, he can assume that there is indeed a bullet in every font and in every keyboard arrangement. This is what we mean by a pseudo-font. The actual location of characters on the output device can be determined by what is most efficient for that particular machine. (It may take too much time to transfer from one type disc or film strip to another every time a period is required, and it may be helpful to have more than one "e" spinning around on the font, in order to accelerate the throughput of the typesetter. The task of the computer program would be to put out a generalized code for an "e" and the typesetting machine itself might be required to keep track of where the "e's" were located and flash the first one that happened to come around.)



In like manner, the assignment of characters to the input device or devices should be arranged to optimize input efficiency. This is not merely a function of locating keys and corresponding characters in positions where finger travel is minimized. It may prove to be most efficient to locate characters where people expect them to be. Thus, although a better keyboard arrangement can be developed—and several studies suggest that the Dvorak keytop assignments may be substantially more efficient than the "QWERTY"—it is the reaction of the average

user which will determine what keyboard arrangement and character assignments should be chosen for the input task.

For our part, we have come to the conclusion that conventional groupings of characters will probably work out best, but how large a group of characters should be assembled together in one "set" is itself an interesting topic for speculation.

Definition of terms

As an aid to the establishment of input conventions, greater precision in our vocabulary would be helpful.

Some years ago Mr. Walter Jessel, of IBM's System Development Division in Boulder, Colorado, prepared a paper entitled "Extended Character Sets in Systems with Photoprinter Output"[1] and presented the following definitions:

1. In a draft dated January 21, 1970.

Alphabet: A set of letters or other characters with which one or more languages are written.

Sign: A letter of an alphabet, a figure, or a symbol standing for one orthographic identity, independent of graphic appearance or size.

Text: A purposeful concatenation of signs.

Graphic: A representation (form, shape) of one sign (including space) contained in a font.

Character: The basic unit of a font; a general term for both "sign" and "graphic," making no distinction between content and form.

Orthographic equivalents: Graphics of different design or size which would not alter the meaning conveyed by text (*e.g.*, a word) in which they are contained if they were substituted for each other. (*Antonym:* **Orthographic contrast.)**

Font: A complete, purposeful assortment of characters for the composition of text in one alphabet, containing one and only one graphic for each sign. (The term **font** is deliberately restrictive to obtain unambiguous access to each member of the set from a single code, or through prefix conventions, from two codes. One font may include upper-case, lower-case, and small cap letters of the alphabet, in additional to figures and symbols.)

Font section: A program-addressable subset of one font, containing 191 or fewer encoded graphics, with optimally ordered code assignments conforming to standard provisions. Not all fonts will contain all sections. Some sections may be common to several fonts.

Graphic escape code: A control character, which, when preceding a sign code, causes that code to represent a different sign. A mnemonic relationship will usually exist between the original sign and that produced through graphic escape. *(E.g.,* Spanish open exclamation point = Graphic escape + Exclamation point.) A graphic escape code may be generated on an output device by a special function key, or by a key whose original function was given up.

This writer is not entirely in agreement with some of the suggestions that Mr. Jessel advances in his paper, but he finds the terminology extremely helpful. For example, the letter *a* is a "sign," whether it is roman or bold or italic, in eight

points or eighteen. On the other hand, a times roman eight-point *a* is a *graphic*. An italic ***a*** and a roman **a** are *orthographic equivalents*. Our major point of difference lies in the fact that we believe a *font section* should consist of any arbitrary number of characters (it could be called a font subset) smaller than a font, and the *graphic escape code* cannot, in our judgment, meaningfully and usefully be followed only by a single character. The mnemonic symbology must allow for greater variety, since the character must presumably be selected from an extension of a basic font section.

Actually, Mr. Jessel doesn't quite come up with the definition we feel to be vitally essential. *Font section* relates to a purposeful selection of particular *graphics. What we need is a term which relates to a purposeful selection of a standard grouping of signs, which we shall call a pseudofont*, or perhaps more accurately, a *standard signset*, since "font" implies graphic characteristics, and the signs we have in mind would have orthographic universality.

Such a standard signset should consist of 256 or fewer signs. Since more signs than this are required to handle scientific, technical and foreign language setting, there must be alternate signsets, and the number of these must be considered to be infinite, since different arrangements of signs would be most efficient (at least for input purposes) for different classes of work. For example, one signset might consist of an arrangement of English, Greek and math. Another signset might present the Greek alphabet without either the English or the math signs.

There will always be some ambiguity in the presentation of signs. Is a superior number the same "sign" as a regular number, or is it merely a different "graphic"? We may worry less about the definition of a "sign" in this context than we would about the method of arranging the keyboard. For certain classes of input it will be most convenient to position superior (and possibly inferior) numbers on the keyboard. Alternatively, one could visualize an entire signset of inferiors and still another signset of superiors. The same thing could be true of inferiors to inferiors, superiors to superiors, superiors to inferiors, etc.

Ideally, we could conceive of four different arrangements of signs and graphics: one for input, in order to optimize the efficiency of input for a particular class of work; a second for output, in order to optimize the efficiency of the output device, given its own peculiar characteristics. A third arrangement of signsets might be adopted by the programmer or systems analyst in his effort to optimize the efficiency of his own software. A fourth signset could be developed for purposes of information interchange. This would contemplate a standard signset, and a certain number of agreed-upon common variations from the standard to accommodate the treatment of scientific information exchange.

There are profound problems involved in this entire area. No signset conventions have in fact been adopted, despite some efforts on the part of certain committees to reach agreement. Neither EBCDIC nor USASCII properly addresses the subject. When it is dealt with intelligently it will be clear that the

differentiation between signs and graphics must be observed, and ultimately we should arrive at a convention which would enable us to compose *all* signsets in any available graphic design (with appropriate default arrangements to access alternative characters that might not be available in a selected type face). This would of necessity assume that the typesetting device at all times had access to its entire repertoire of graphic images, and it almost certainly would imply a third-generation typesetter.

Some special considerations

One problem has to do with the representation of *small caps*. Some type fonts offer capitals, lower-case letters, and small caps as well. The use of small caps needs to be specified at the time of input, and these characters need to be accessed on output. In the internal program, their width values need to be stored and accessed, and occasional words might have to be hyphenated. To add small caps to a signset might be misleading, however. They are to be differentiated as graphics rather than signs. The present writer believes that the best way to handle small caps is to consider that, as signs, they do not differ from lower-case characters, and to create a special pseudofont within the program which places small caps in the lower-case positions. The program does not, in fact, need to "know" that these are small caps rather than lower-case characters. The output program would, of course, have to access them.

ALL CAPS,
SMALL CAPS, &
CAPS & SMALL
CAPS

Another area of difficulty has to do with *ligatures*. It would be unfortunate if the user were required to key ligatures as separate "characters." This would present a special problem to the hyphenation routine, and would also make it necessary for the person performing the input to know whether or not a particular font did or did not offer ligatures. The better solution is not to include ligatures in the signset at all, but to rely upon the computer program to access ligatures wherever they are required (so that their widths would be stored in the character tables) and to output these characters in the driver program. Consequently, ligatures would not appear on the keyboard, as signs, but only on the typeset output, as graphics.

fi
fl ff
ffi ffl

Fractions may be treated in the same way. The operator would key "1/2" for "one-half" and the program would generate "½" if a one-piece or stand-up fraction (sometimes called an "en" fraction) is available for the output device.

Accents, too, present special problems. Is an accented character a "sign"? It may be convenient to think of the accent as a sign, and the character below the accent as another sign. But accents are sometimes provided as "one-piece" characters, and at other times accented letters are created by the use of "floating" accents. In the latter case the program determines whether the width of the accent is greater or less than the width of the character, and "puts out" the wider width, centering each in relation to the other. In the case of an accent over a lower-case "i" the program has to be "bright" enough to access a "dotless i" instead of a regular one.

In Europe, some operators would prefer to have distinct keys for accented characters. In English-speaking countries, it is usually easier to key either the accent and then the character, or the character followed by the accent. This is simply a problem of input convention. As far as output is concerned, the program can either access a one-piece accented character or generate a floating accent over a conventional letter depending upon what is available. In fact, the program should be "clever" enough that the operator should not be required to stop to think whether he requires an accent over a capital letter, a small cap or a lower-case character. The program should find this out and should fetch the accent which is appropriately positioned, just as it should be able to fetch the accent which matches the type face—whether roman, italic, or bold, for example.

Œ œ

ß ç

áàâäã

ÁÀÂÄÃ

göras på

Grâce

même

åsamka

système

Sølvgade

Another problem is posed by the possible use of *"digraphs"* or diphthongs, such as *Ae,* or *Oe.* These one-piece characters cannot be combined by the computer program since they occur only occasionally. Consequently, they must be considered as "signs" and input accordingly.

The reader may be interested in a keyboard layout which the author helped to develop for the National Geographic Society. This is particularly interesting because it poses most of the problems we have touched upon, only, perhaps, "in spades." There are the following accents: *acute, grave, breve, macron, umlaut, circumflex, cedilla, klicka, tilde,* Polish hook, Polish slash, barred *D* (upper- and lower-case), some special Arabic reverse apostrophes, and single dots, lines or hooks under such characters as *s, t, h, e, d* and various others. In addition, there are, of course, the reverse (opening) exclamation and question marks of Spanish, the chevron-like quotation marks for French, the lowered opening quotes for German, as well as a series of digraphs for *AE* and *OE* combinations in caps, small caps, lower-case and combinations thereof.

In this instance the keyboard offers a central keypad, and two supplementary keypads if we disregard those used for editing and processing commands (such as *insert, delete, move, copy, execute, etc.*). It was decided that all accents would be keyed as if they were floating accents with the exception of barred *O, L* and *D* (upper and lower case). We had available to us a "supershift" mode as well as a shift mode. The shift and supershift modes change the bit configurations. One key can represent four characters. That key, when the shift *or* supershift, or shift *plus* supershift keys are held down, will cause codes to be recorded in entirely different patterns.

We created a different signset layout for combinations of caps and small caps, and the small cap digraphs of *AE* and *OE* also appear on this alternate signset arrangement. In like manner, we agreed that the alternate signset layout would imply the use of "Oldstyle" (non-aligning) numerals rather than the more conventional base-aligning figures.

The layout also contemplates the use of three different kinds of hyphens: a hyphen to be input for compound words, a "discretionary" hyphen to be input to assist the program, where necessary, in breaking words, or ordering that words

Atex terminal keyboard layout customized for National Geographic

™ √	FS ⩾	Δ ÷	□ ∞	■ ×	© ′	® ″	▼ ◀	▲ ↑	‡ ▶	| ◀	¶ ▶	< {	° <	- ʒ	en em leaders	START
⅛ +	¼ ⩽	⅓ ±	⅜ ★	½ ☆	⅝ Æ	⅔ Œ	¾ μ	⅞ ↓	† æ	∮ œ	£ \	> }	= >	EN i	MERGE	COM.
																END COM.

Supplementary pad

/ ′ ··	ˇ ″ ~ ◡	° % ◡ ◡
◡ ◡ / ◡	– \	— ÷ △ ÷

Supplementary pad (located on left)

EM	! 1 1	@ 2 2	# 3 3	$ 4 4	% 5 5	¢ 6 6	& 7 7	* 8 8	(9 9	) 0 0	BLR [—]	- MH - DH	RUB-OUT		
QUAD MIDDLE	Q q	W w	E e	R r	T t	Y y	U u	I i	O Ø o ø	P p		QR	QC		
SHIFT LOCK	A Å a	S s	D Đ d đ	F f	G g	H h	J j	K k	L Ł l ł	: ;	' '	QUAD LEFT			
SHIFT	Z z	X x	C c	V v	B b	N n	M m	, ,	. .	? /	SHIFT				
Supershift key is at left.	SPACEBAR														

Keyboard layout:

S SS

U US

where:
S = shift
SS = supershift
U = unshift
US = unshift supershift

not be broken (when the discretionary hyphen appears at the beginning of the word), and a "machine" hyphen. The latter is generated by the program when words are split; it is never input by the operator, but it is required to be on the keyboard in the event the operator wishes to command the computer to institute a "global search" for every word hyphenated by the "machine."

The use of the supplementary keypads may also be of interest. Here we departed from the notion that fractions should be created by the program, and the common fractions do in fact have keys of their own. All others would be generated. Ligatures are of course not included since the program will create them where needed, and the kerning of characters is implemented by table look-up, although special typesetting commands can also be given for space-reducing effects.

Except as indicated, the alternate (SMALL CAP) signset arrangement duplicates the first or standard signset layout. Changes in point sizes and type faces are introduced by other keys, not shown.

Although this may appear to be a very complicated solution, the problem is a complicated one. Nevertheless, every single one of these characters in the standard signset layout can be represented by a unique eight-bit code with enough codes left over to describe characters which need not appear as signs (*e.g.*, ligatures) and to give other instructions to the output device or devices.

It will be seen that we have tried to use the National Geographic keyboard layout to represent signs, and not graphics. Other keys we have not shown would call for different "modes" or type faces, which would imply the graphic images used to convey these signs—whether Caledonia, or Times, italic, bold or roman. Similar commands would signal the point sizes in which we were to compose the type. Even with these conventions, however, the keyboard is rather large, in terms of numbers of keys—larger than one might wish. A more conventional keyboard may be desired. Yet even that keyboard should have the potential to describe not only the signs we have placed on the National Geographic keyboard, but many more as well. It is for this reason that *escape* codes are often used. An escape symbol will signal the fact that the user wishes to shift to a different sign or to a different graphic, or both. In a sense, the supershift key which is located on the left of the National Geographic keyboard accomplishes this by enabling the keyboarder to access two additional code groupings (in the shift and unshift positions) so that every key can have four meanings. But in this instance you are not escaping from the basic 256-level code configuration: There is still a unique code for every sign (although not for every graphic).

It is also possible to escape from the ASCII or EBCDIC representations themselves (or most of them) so that each code can stand for an entirely different signset—Cyrillic or Hebrew, chemical symbols, astronomical signs, representations of the four suits in bridge, or whatever else one might require. If this is done, then the program must know whether or not to subject the code stream to the hyphenation program. Unless told, it may think that it is working with English language words when it is not in fact doing so. But the program could be so conceived that when one "escapes" to another signset (as one might be conceived for the French language, for example) a different hyphenation program (French logic hyphenation) could be invoked.

The role of the typewriter

For English, and perhaps for other languages, the typewriter will probably retain its role as a frame of reference for keyboard design. Although some—perhaps most—keyboards intended for use in connection with composition provide auxiliary keys, the typewriter can indeed be made to serve. For example, since the typewriter cannot express a fixed space value, input into a computer system can be translated to recognize that the combination "(em)" shall be translated as an em space, and "(en)" as an en space. In like manner, "(ql)" can mean "quad left," "(qc)" can stand for "quad center," and so forth. Even special characters can be described by mnemonics. One could use "(de)" for a degree symbol, "(da)" for a dagger, "(mu)" for a Greek letter *mu*.

It is obvious that more keystrokes are required to provide the full signset desired, and still additional keystrokes will be necessary to state the typographic codes such as *change font, change measure, etc.*, but these and all other commands—even to the most complex—can surely be described by some combination of unique keystrokes, and translated by the input program. For a more de-

tailed explanation of this subject, the next chapter, which deals with optical character recognition devices, will furnish examples and illustrations.

In other words, what is involved, when the keyboard is limited, is recourse to a kind of "keyboard longhand" which requires a more roundabout way of indicating the desired signs and functions. For some purposes, the simple typewriter keyboard can provide a sufficient choice of signs and symbols. For more complex work, or for more efficient input, a somewhat extended keyboard may be desired, since it reduces the number of keystrokes. On the other hand, one can carry the "key for every purpose" concept too far. It may, in fact, be just as easy to type "(egr)" for *"e accent grave"* as to type the letter "e," followed by the use of a special key, somewhat removed from the standard keypad, which interrupts the operator's rhythm. This is obviously a matter of human engineering. It may be difficult for the operator to remember what the combinations of letters are supposed to be to describe a particular sign or function. The more "mnemonic" the combinations, the easier they will be to use. The more uniform the input, for a specific requirement, the easier it is for the operator to retain in his or her memory the mnemonic or other keystroke combinations which are specified.

Experience seems to indicate that certain kinds of instructions, being somewhat complex, can best be done with special keys. For example, Automix (AKI) offered a 70-key layout, an 80-key arrangement, another for 100 keys, and still another which provided 120 keys. Similar choices are provided by some other manufacturers. Moreover, certain combinations of keys are engineered to discourage input mistakes. The AKI CIM-100 provided a key for "line length," and another for "point size." If one depresses either of these keys, all keys are locked except for the numeral and punctuation keys, and this condition prevails until the operator has keyed the requisite number of digits to complete the command. In like manner, AKI's PCI-120 provided a special keypad or key bank to the right of the main body of keys. It is this right-hand pad which is used for most commands, and the main board is locked until a proper combination of keys has been struck on the right-hand pad.

So far as we have been able to ascertain, when production keyboarding for composition takes place, users seem to prefer more keys rather than fewer, believing them to be more efficient since they require fewer keystrokes to implement commands and to describe the desired signs, and they also have safeguards of the sort we have just mentioned built into them.

Such keyboards also provide one other feature which is of great interest. That is the use of "programmed" keys. Special keys can be provided which will output multiple codes with one keystroke. Some such keys are pre-programmed at the factory for a particular user. If that user wishes to drive a Mergenthaler V-I-P typesetter, for example, he may want one key which will furnish to the typesetter the multiple keystrokes necessary to indicate a line length command, a font command, or a "change point size" command (in each instance to be followed by the "argument" or values specified). The manufacturer will install in his keyboard

one or more "PROMS"—programmable read-only memories—which have been programmed at the factory to respond to a keyswitch by outputting the desired codes. Beyond this, many keyboards offer additional keys which are programmable by the user himself, even for the individual job. If the keyboard contains "RAM" memory (random access memory) the user can store formats, either directly from the keyboard or "loaded" into the keyboard from an optional paper tape reader. If the keyboard contains eight such keys (designated in our illustration of the PCT-120 by "Mem 1," "Mem 2," *etc.*, for "memory" keys, the user can store up to 64 keystrokes in RAM to be accessed by any one key; or, if he wishes to differentiate between upper and lower case when he uses these memory keys, he could store 16 formats of 32 keystrokes each. Such keys offer a form of "keyboard shorthand." Not only can they be used for typographical commands, but also for repetitive phrases, such as "Merrill Lynch, Pierce, Fenner & Smith Inc."

Hard copy, soft copy, and "blind" keyboards

At one time a debate raged as to whether hard copy keyboards or "blind" keyboards were more efficient. The hard copy keyboards produced an output, as does a regular typewriter, either by activating striking bars, or through the use of a typing element or "golf ball." Consequently, except for certain functions which could not be represented by the typewriter keys,[1] the operator could tell when a mistake was made, and "rub out" the incorrect keystrokes. But it was argued that such keyboards were less efficient because the hard copy distracted the operator and the operator's rhythm was interrupted by the need to wait for the carriage return. Consequently, many favored the use of a "blind" keyboard which merely punched holes in paper tape and provided the operator with no visual feedback.

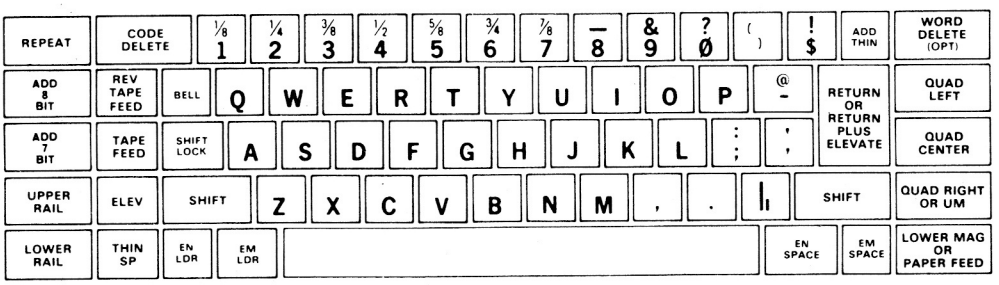
1. Em, en, and thin spaces, for example.

But "soft copy" keyboards soon came into being. These provide a representation of the character on a moving display (sometimes called "self-scan" display), much like the manner in which the Times Square illuminated signboard carries the news. Such self-scan devices generally show either the last 32 or the last 64 characters keyed, and the best of such units will represent every single character by an appropriate visual image. When these displays are used there is no delay for a carriage return, and if the operator is distracted by the visual representation (which we think unlikely) the display can be turned off. Nowadays, as an alternative to a self-scan display, a CRT display may be provided, and some cursor editing capabilities may also be present. These will be discussed in our chapter on Video Terminals.

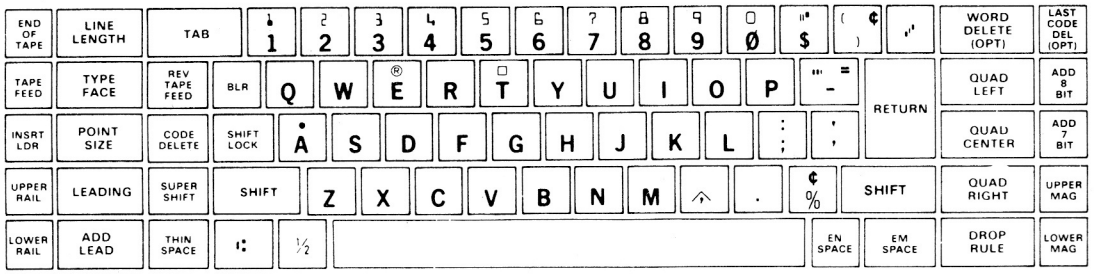
Counting keyboards

While many keyboards are used to create input into a computer system, where the computer performs hyphenation and justification and makes end-of-line decisions, or directly to an "intelligent" typesetter, where the "front-end intelligence" of the typesetter makes such decisions, there is still some demand for "counting" keyboards. Such keyboards need to be capable of accepting tables of width values for the specific fonts to be set. Usually such width values are provided by pre-programmed read-only memories. (One such keyboard provides

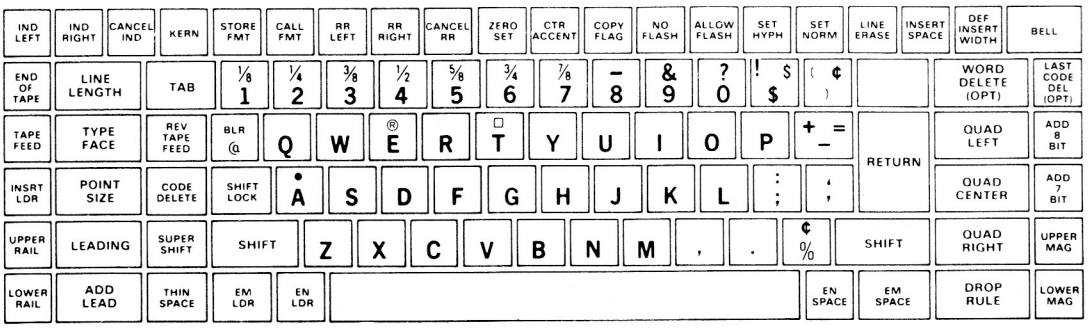
CIT-70 Keyboard Layout



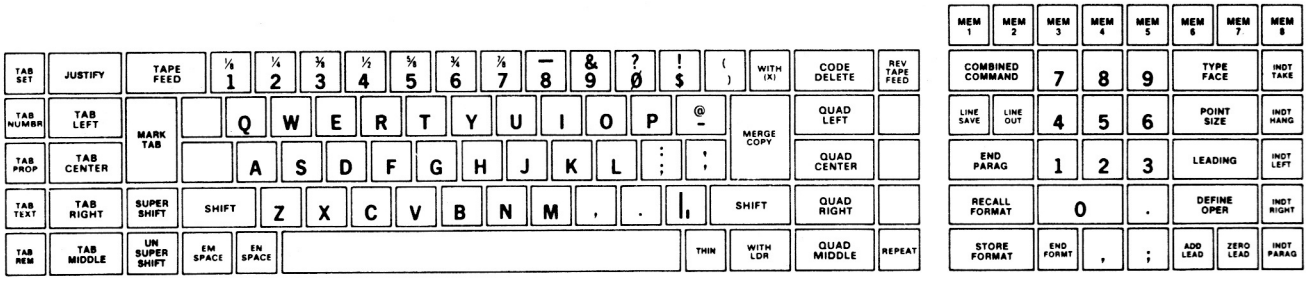
CIM-80 Panel and Typical Keyboard Layout



CIM-100 Panel and Typical Keyboard Layout



PCI-120



sockets so that up to 48 ROM chips, representing 48 fonts, can be plugged in at one time. Or an off-line storage box can be plugged into the machine, enabling the operator to access as many as 255 fonts, which are stored on magnetic tape, and read into RAMS within the keyboard.) Counting keyboards are still useful for the highest quality composition, especially where the operator relies otherwise on the front-end intelligence of the typesetter. Such keyboards are also useful for preparing correction lines, where the "computer" did not hyphenate properly, or some other mistake arose, and it is not useful to reprocess the entire "take."

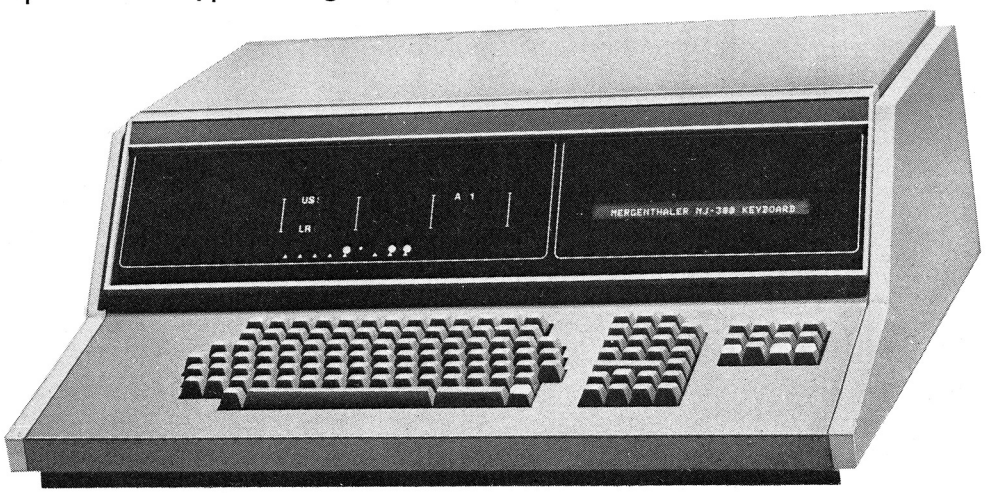
Direct-entry typesetters have also gained in popularity, and they are often operated in a "counting" mode, although "automatic" or "semi-automatic" options are available, freeing the operator from the need to make some (or all) line-ending decisions.

Relation of input to function

It should be clear that efficient production requires a careful study of various input alternatives, with as much customization as possible to meet the requirements of the individual user, given his output device, his "system" concepts and the kind of work he produces. On the other hand, for some purposes, generalized input is desired, which can be processed in a variety of ways. It is even possible for the customer to produce his own input and to pass it on to a supplier for processing. At present this would require extensive communication between buyer and vendor, so that all of the typographic and programmatic conventions and requirements will be met. There is some possibility, however remote, that in the future certain standardized and universal input conventions will be developed. As we should be able to understand by now, these will have to relate to the signset conventions as well as to those which instruct the typesetter as to format, type faces, line lengths, indents and the like.

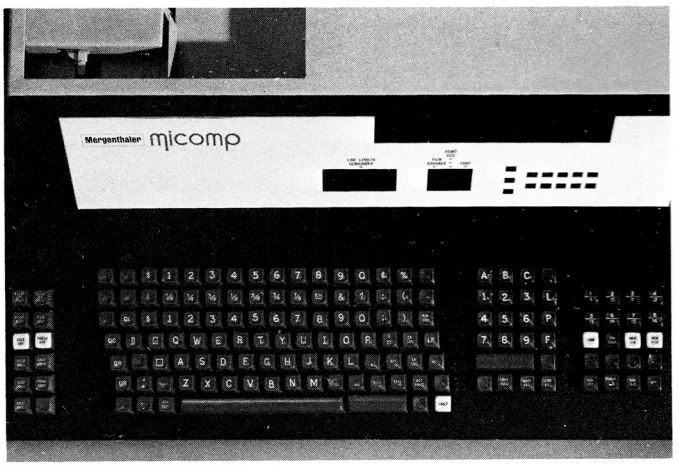
Review Questions:

1. Why does a composition system's "internal language" usually differ from both the input coding and the output driver language used?
2. What is meant by the term "position keyboarding?"
3. Devise some concrete examples of possible pseudofont input coding and output mapping for a few pi characters.
4. How would you differentiate between the terms "graphic" and "sign?" What is a "standard signset?"
5. Give a few examples of graphics which it would be preferable **not** to treat as separate signsets.
6. How would you recommend that accents be treated in terms of keyboard layout and design? What about digraphs?
7. How might "programmed" keys on an input keyboard be used?
8. Assuming you had only a standard typewriter keyboard with which to work, describe the mnemonics you would use to access ten different pi characters and/or typographical commands.
9. What do we mean when we use the terms "blind keyboard" and "counting keyboard?"



Left: Mergenthaler's NJ-300 input keyboard is a non-counting device. But it is not a "blind" keyboard since it provides visual feedback to the operator by means of a 32-character display screen.

Right: A counting keyboard such as Mergenthaler's Micomp assists the operator in making his or her own line-ending decisions by providing two kinds of visual feedback. A 32-character display shows the text as it is being entered. In addition, font width information is loaded into the device's memory enabling the keyboard to provide a continuous countdown of the number of units remaining in the line as each character is entered.





Left: Varisystems' VariComp 2000 is a "blind," non-counting, paper tape perforating keyboard. It provides no hard copy, nor any other form of visual feedback.

Right: Some input keyboards incorporate limited format storage and editing capabilities. The Memory Text non-counting keyboard, which is a product of G.O. Graphics, allows the operator to correct keystrokes as they are generated by overstriking, inserting or deleting characters within the unit's 32-character display memory.



17

Optical Character Recognition

DISCUSSION OF THE ROLE of Optical Character Recognition is relevant for several reasons: First, because OCR has played and will no doubt continue to play some role in the composition process; second, because a consideration of OCR will give the reader a clearer understanding of the way in which input which can encompass no more subtlety than can be expressed through the 86 characters represented on a typewriter can nevertheless be adequate to meet the most demanding typesetting requirements. In the third place, a consideration of OCR begins to introduce us to the "systems aspects" of composition, wherein we start to put together the ingredients which comprise a viable technique for translating the initial manuscript into typeset pages.

History of OCR

Experiments with optical code recognition and optical character recognition go back further than most people realize. Patents on an OCR device were filed by Hiram Goldberg in 1912. Considerable research was done both by Columbia Broadcasting System and by the Radio Corporation of America during the 1930s and 1940s, largely as an outgrowth of work on television systems. In the late 1940s, a company called Intelligent Machine Corporation began to work on a machine to read numeric information printed on credit card slips. Its first commercial reader for this purpose reached the market in 1954.

It cannot truly be said that the world was waiting breathlessly for OCR readers. By 1970, sixteen years after the first Intelligent Machines model appeared on the market, it has been estimated that something on the order of 1,000 machines were in use worldwide. (The exact figure may depend upon a precise definition of what is meant by an OCR reader.) By now[1] that figure has grown to well over 4,000, with more than one-third being used in typesetting applications, and an increasing number being sold to augment word processing systems.

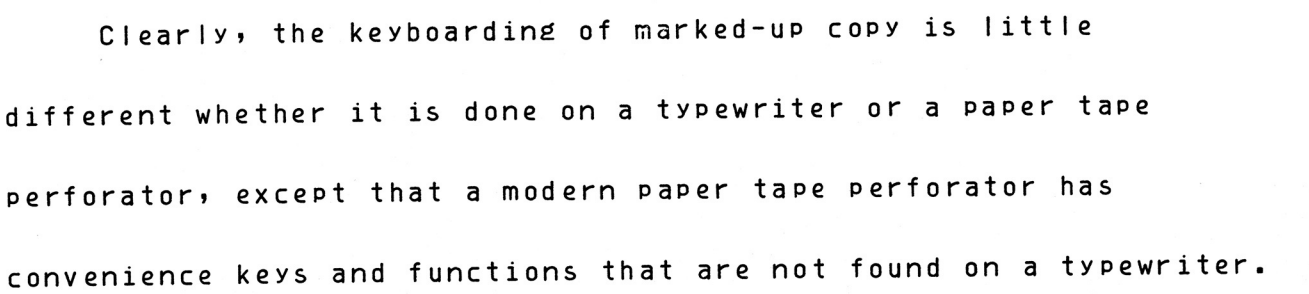
1. That is, in early 1979.

Clearly there has been a significant increase in the number of machines over the past decade. And by far the largest portion of that increase has been in the

graphic arts industry. Before 1971, there were very few scanners used for typesetting applications. But in the early to mid-1970s at least three times as many OCR readers were sold for graphic arts applications as for all other uses combined.

These figures clearly suggest that there was a dramatic change in the use of OCR readers in the graphic arts field after 1970. The OCR reader was transformed from an expensive and esoteric device into a production tool. And, in the process, typesetting applications became the largest single market for OCR devices, and the largest impetus for the renaissance of interest in OCR generally. Were it not for the changes which took place in 1970 and 1971, the subsequent proliferation of OCR would not have occurred.

The first scanner used specifically for typesetting input was a unit built by Recognition Equipment, Incorporated, and installed in 1965 at Perry Publications' newspaper plant in West Palm Beach, Florida. Two REI scanners were eventually used there to process both news copy for various Perry newspapers, and for commercial work as well, until 1969, when John Perry lost interest in the newspaper business and sold off his papers and scanners. Essentially, all that remains now of Perry's pioneering efforts is the "Perry" typewriter font, developed for use in West Palm Beach and modified slightly for the CompuScan reader. It is still one of the most widely used OCR fonts.



The Perry font.

Aside from the Perry installation, most efforts to use OCR input for typesetting during the 1960s were focused in one of two directions. First, there was the application of OCR essentially as an alternative to paper tape perforators. Copy was re-typed for OCR input, then fed into a scanner. The OCR readers for this purpose were generally large and expensive: the REI unit, Control Data machines, IBM, and Scan Data readers. The size and cost of these readers dictated either an exceptionally large volume of work or a service-bureau type of operation in which the costs of such expensive equipment would (hopefully) be distributed over a number of users.

The primary advantage of scanned input, according to its proponents, lay in the fact that typed input was faster and/or cheaper to produce than input created on paper tape perforators—especially as "relatively untrained" typists could be used, perhaps even in a "cottage industry" situation in which the typist is paid only for work produced, and need not be paid at all if there is no work to perform.

More recent developments in both OCR readers and alternative technologies have changed the picture somewhat. Scanners are now far cheaper and much more powerful, but paper tape, magnetic tape, or, more likely, key-to-disc input keyboards and video terminals are also far more effective than they once were. The merits of re-typing copy specifically for OCR input are less heatedly debated today.

Scanning typeset documents

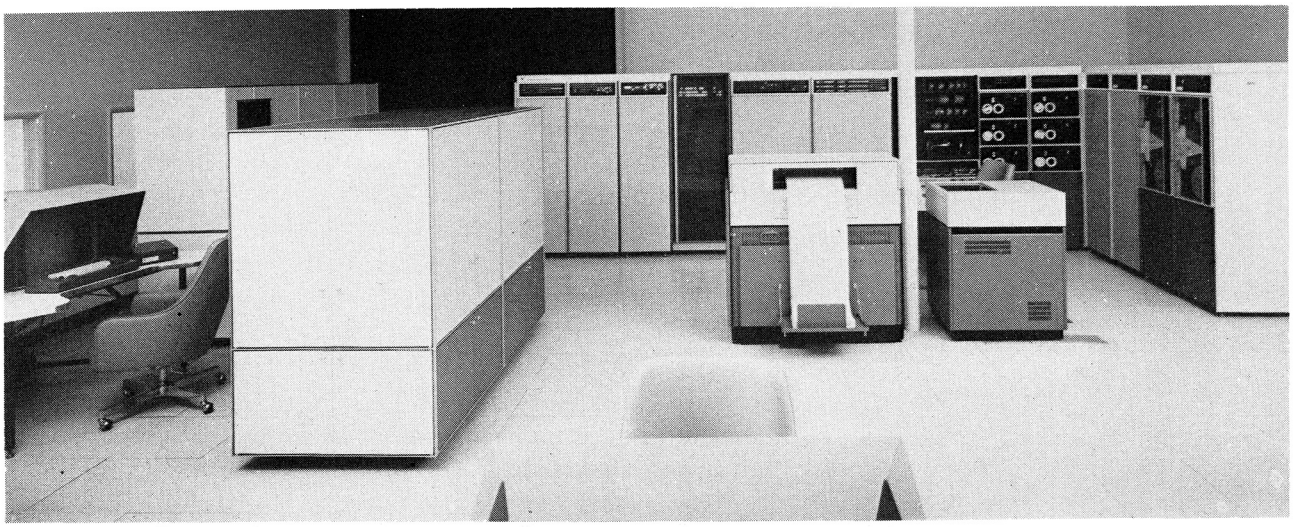
The second approach to OCR during the past decade was to use OCR readers to "scan" copy which had already been typeset and printed. Many jobs which were kept in hot metal and updated year after year could be better handled as computer-stored data files. The problem has been that the "capture" of such jobs for computer processing required rekeyboarding and reproofreading all of the text which was "stored" in metal. Other jobs, such as a Bible, a text book or a training manual, might already have been typeset and printed and the metal type subsequently destroyed. In both cases it would be quite useful to "read" material which had already been typeset and printed so that it could be converted into a form which was processable by a computer.

Naturally, reading *typeset* material requires a far more sophisticated OCR device than does reading *typewritten* material. In the first place, the scanner has to cope with proportionally-spaced rather than monospaced type. Secondly, it has to be able to read a variety of fonts—none of them designed with any thought for OCR input. In the third place, the quality of the printed output is often uneven, smudged, and the pages to be read are less susceptible to the use of an automatic transport or feeder.

Several OCR readers have been built for this purpose. The most widely used have been the Scan Data scanners, of which perhaps three or four were sold principally for the reading of typeset input. Two other readers were built for this purpose, although both were more expensive than the Scan Data model, and only a few have ever been sold to a commerical customer. The first of these was the CompuScan Model 370 and the second one was the Information International GRAFIX I scanner. Both of these are omni-font readers which have heuristic (self-learning) programs which allow them to acquire new fonts by actually reading them. In 1978, the less expensive Kurzweil reading machine was offered to the market, and its introduction may revive interest in the "reading" of printed matter. This device also offers simulated voice output for the benefit of the blind.

When presented with a new document and a new typeface which it has not encountered before, a scanner of this type will stop and ask its operator to identify each character it reads. In this way, the operator will "teach" the scanner to recogize a new type face or a new version of an old type face.

Even with "omni-font" recognition logic and microfilm document transports to simplify the task of bringing the text to the scanner, the problems of reading already-typeset material are indeed formidable and devices with such capability



Information International's GRAFIX I Automated Data Input System includes an "omni-font" scanning capability. Below is a sample of a document which can be "read" by the GRAFIX I system. It includes both computer printed information (smaller letters and numbers) and hand-printed data.

RD8(GT) TAKE-ON SHORT TERM BENEFITS | TECH. DETAILS | DATE OF ISSUE | LEDGERS LINE QUOTA PAGE

PLEASE FORM CHARACTERS AS SHOWN HERE- 1234567890AB

LEAVE COLUMN BLANK IF NOT APPLICABLE

0	1	2	3		4	5	6	7	8	9	10
IF NO RECORD SHEET HELD OR IF AN ERROR CANNOT BE CORRECTED- ENTER 1	GBU LOCATION- SERIAL NO. OF LAST LOCAL OFFICE	DATE OF LAST SB CLAIM (OR FOR WOMEN, SB/MB CLAIM) MONTH/YEAR	ENTER ONLY IF LAST UB CLAIM IS AFTER 1st JAN 1973 ADP PREFIX	UBO CLAIM PAPERS LOCATION	DATE OF LAST UB CLAIM MONTH/YEAR	ABROAD CODE OPEN NOTE -1 CLOSED NOTE -2	CIVIL SERVANT -1 H.M. FORCES -2 (OPEN NOTES ONLY)	OVER-PAYMENTS LOCATION CODE NIO -1 UBO -2 CPB -4	OVER-LAPPING BENEFIT CODE DWP -1 IDB -2 GA -3 (NEGOTIAM) -4	IF BENEFIT CONTROL INDICATOR TO BE SET ENTER 1	IF NO TAKE-ON REQUIRED INDICATOR TO BE SET ENTER 1
		YM104264C		BLA PF	BLAKE	PAUL		FRANCIS			
	34827	12/73	B	72-520	9/72	2		3	4		
		KP067616A		NEW DH	NEWHOUSE	DONALD		HENRY			
1	20089	5/74	C	55-667	6/74	1	2	4			
		AS330617A		LEF FE	LEFF	FRED		ELLING			
	56084	4/74	A	61-032	10/74	2		1	1	1	
		YS390037D		BOY RP	BOYER	RALPH		PERCY			
	22985	5/69	D	60-854	12/73	2	1	1			
		YK502537D		HAN ER	HANSELL	EDGAR		RICHARD			
1	34576	2/72	C	48-056	2/74	1		2		1	
		KR067763A		SNE PN	SNEED	PAULA		NANCY			

are extremely expensive. Thus, with the exception of some on-going work on the GRAFIX I—applicable to large documentation activities of a few major corporations and government—and the newly-emerging Kurzweil technology—the emphasis in the graphic arts has shifted away from these larger, more expensive machines used to read previously typeset input desired primarily to create a "data base," to a lower-cost breed of OCR readers developed first by ECRM in 1970, and then by other vendors. It is on this new breed of reader that we will now focus our attention.

The ECRM concept

A company called ECRM[1] (possibly standing for "Electronic Character Recognition Machine") was founded in 1969 with the purpose in mind of building an OCR reader which would sell for under $100,000, and with enough flexibility to handle typesetting output from typewriter input. The REI, IBM, CDC, Scan Data and CompuScan units we have mentioned heretofore sold for prices ranging between $300,000 and $900,000—a decided barrier to widespread use in an industry dominated by relatively small firms.

1. ECRM has now been acquired by AM International.

Equally important, ECRM conceived of a scanner with different functional specifications than earlier models. The ECRM scanner from the outset was designed to be a machine to be used in newspapers. Reporters were to type copy in a scannable font. *That copy would then be edited,* and then passed on to the scanner. Similarly, classified ad takers would type directly onto forms which could be read by the scanner.

The key elements in the ECRM scheme were:

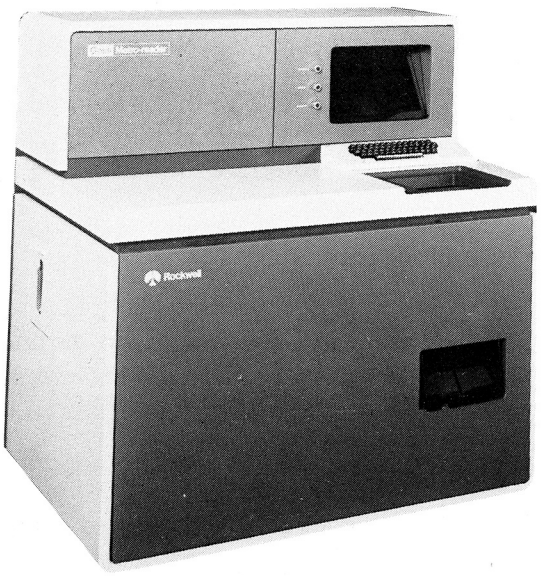
- Material created at the newspaper would be *captured at the source in machine-readable form.* Therefore, copy never has to be rekeyboarded when it reaches the composing room. (And, presuming the original copy is "clean" and that the scanner is accurate enough, if it never has to be rekeyboarded it never has to be re-proofread and re-corrected.) Hence, most of the time-consuming and labor-consuming steps in the composing room are no longer required. Copy can be processed through composition considerably faster than was previously the case. Most of the composing room jobs for keyboard operators and proofreaders were therefore no longer needed. In effect, the compositor would be replaced by a machine. Or, more precisely, by two machines: an OCR reader to read and interpret copy, and a computer to make end-of-line decisions.

- Copy which is "captured" in machine-readable form is created on the ubiquitous and inexpensive typewriter. No expensive computer terminals for tape-producing keyboards are required and anyone with basic typing skills can learn to produce acceptable input.

- The "paper flow" within a newspaper would be preserved. Editors would still work with the familiar typewritten copy, rather than with unfamiliar devices such as video terminals.

This latter requirement, in particular, imposed a new burden on the OCR reader, since now, for the first time, it was not sufficient simply to read the characters on a page. The scanner also had to generate string explosions from "copy mark-up" commands. But what is even more difficult, the scanner had to be responsive to at least light editing changes—insertions, deletions, replacements and corrections.







Above left: ECRM began the influx of low-cost OCR devices into the graphic arts market. Their Autoreader makes use of a Digital Equipment PDP-8/e minicomputer. *Above right:* MGD's Metro-reader was originally developed by Cognitronics. If the scanner logic fails to recognize a particular character, it will either make its "best guess" or stop on that line, allowing the operator to review the sequence on the unit's VDT screen and key in the proper character string at the keyboard. *Left:* AM VariTyper markets the desk-top Scan/Set (which was designed by Datatype, Inc.) to the graphic arts market. It reads Optical Bar Coded-material at a rate of about 45 seconds per page. *Below left:* Hendrix Electronics' OCR-2 will perform basic functions involving "post-recognition logic." *Below:* CompuScan's Alpha scanner also contains post-recognition logic.





Interest in the ECRM approach was evident immediately. Other manufacturers jumped in: first CompuScan, then Cognitronics (Graphic Systems, Bobst and MGD), then Hendrix and Compugraphic (Dest), as well as manufacturers of optical *code* reading devices: Datatype (AM International) and Taplin (now called Context). Context also supplies an OCR reader as well as an OBR (Optical Bar Code Reader).

In the first six years after the first ECRM production unit was delivered, at least 2,000 relatively low-cost, graphic-arts-oriented OCR or OBR page readers were sold. Although the ECRM concept was the dominant one, not all OCR or OBR readers for the graphic arts industry are used in the precise manner that ECRM visualized. Alternative conceptions relating to the use of OCR have also gained currency.

OCR vs. on-line video terminals

At about the same time that ECRM was getting off the ground, another approach to newspaper production was gaining in impetus—that of "capturing" copy in machine-readable form by means of computer terminals—most frequently video terminals—and then performing all subsequent editing and copy handling by means of video terminals on-line to a central computer data base. Many such systems have now been implemented. Systems combining OCR (for input) and video terminals (for editing) have also been implemented. This approach, it has been claimed, offers some of the cost advantages of OCR input with the editing flexibility of video terminals.

OCR readers, particularly the newer, lower-priced models—now selling for from $35,000 down to perhaps $15,000—have also found their way into commercial (non-newspaper) installations. Here they may be used in a quite different manner than in newspapers. By and large, commercial typesetters do not create the material which they typeset. The commercial typesetter, then, must either train his customer to create input for his OCR reader, or must *re-keyboard* the material before it can be scanned. This represents an entirely different and less compelling set of economic circumstances than those that have presented themselves in newspapers. Nevertheless, there are circumstances under which OCR can play a role in the non-newspaper section of the graphic arts industry.

What is OCR?

The initials "OCR" can stand for either optical *character* recognition or optical *code* recognition, with the distinction being that *character* readers read "human readable" type fonts while *code* readers read a bar code image printed *below* the human readable characters. Sometimes OBR is used to describe the latter devices. Except for recognition logic, code readers behave very much the same as character readers.

The function of OCR readers is to convert the alphanumeric information on typewritten documents into electronic bits and bytes. We almost said into

"machine readable" form, thinking of a magnetic tape or of paper tape, for example, as "machine readable." But because of OCR, the typewritten document itself becomes machine-readable.

An OCR reader, whether it is reading codes or characters, consists of three basic elements: the paper transport, the "reader" or scanning mechanism itself, and the recognition logic. To these, the devices we will discuss in this chapter generally offer a fourth element, which we have termed *post-recognition logic.*

The paper transport

The readers with which we are concerned are all *page* readers. That is, they will read standard 8½ x 11″ or 8½ x 14″ sized pages, and perhaps some other sizes as well. They are not document readers in the sense that they were designed to read post cards or department invoices.

The function of the paper transport is to select sheets in sequence from an input stack and to move them, one at a time, into the scanning area of the machine. This should be done without double feeding, since the material to be read is often consecutive. Once read, the pages are removed and deposited in a bin. In some machines the paper transport may also serve the function of advancing the page to be read, one line at a time, in front of the "read" mechanism. In others it may also actually sweep the line of type to be scanned past the read head.

After the sheet has been read, it will be deposited in the output bin. Most OCR readers have only one bin. A few have two bins, and the machine logic will direct to the second bin any sheets which have been rejected as unreadable or presenting problems.

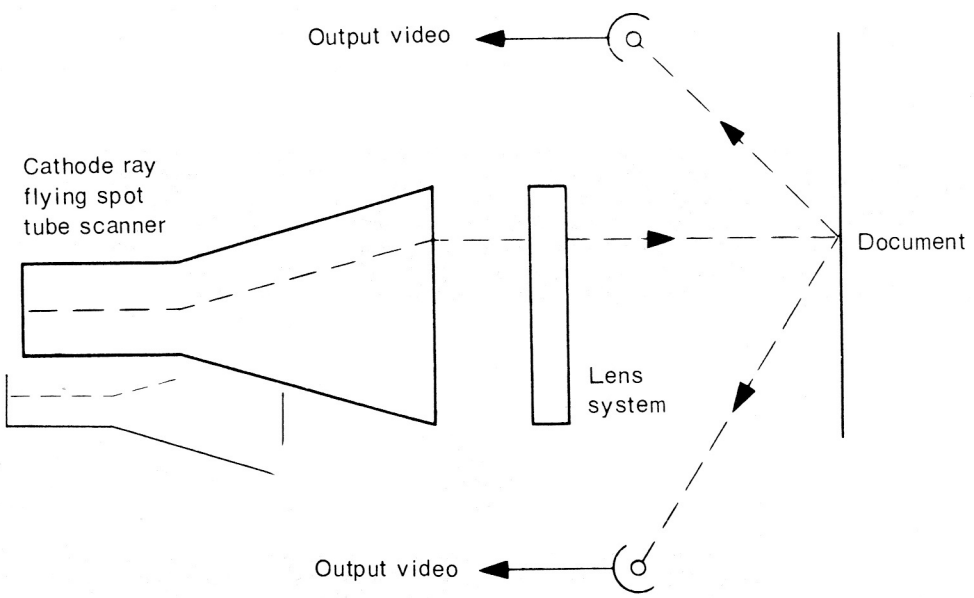


ECRM feed mechanism

The paper transport mechanism is one of the most difficult and costly aspects of scanner design. As anyone with experience with office copiers knows, even Xerox, with all of its resources, has not been able to build a completely reliable and foolproof paper transport. Paper transports can account for something like 50% of the cost of the OCR unit, and 80% of the trouble, and because it is the mechanical portion of a basically electronic package, the paper transport mechanism is the portion least amenable to cost reduction.

The "scanner"

The central operating part of an OCR reader is the scanning mechanism for the "data capture." The function of "data capture" is to reduce the images on the typewritten page to black and white patterns which can be analyzed by the recognition logic. This is usually done by measuring on a point-by-point basis the amount of light reflected from the surface of the paper to be read. The most commonly-used method of deciphering the images on a page has been that of "scanning" the page one typewritten line at a time with a single spot of light and recording the variations in intensity of reflected light as they are observed or detected at a series of closely spaced points across the line. One method of doing this is the "flying spot" scanner. A cathode ray tube produces the scanning beam. The spot of light which results when the electronic beam in the CRT strikes the phosphor-coated face plate on the front of the tube is projected through a focusing lens onto the paper to be read. By deflecting the beam in the tube, the scanner can "walk" the spot of light across a line of copy. One or more photocells "read" the amount of light reflected on a point-by-point basis.



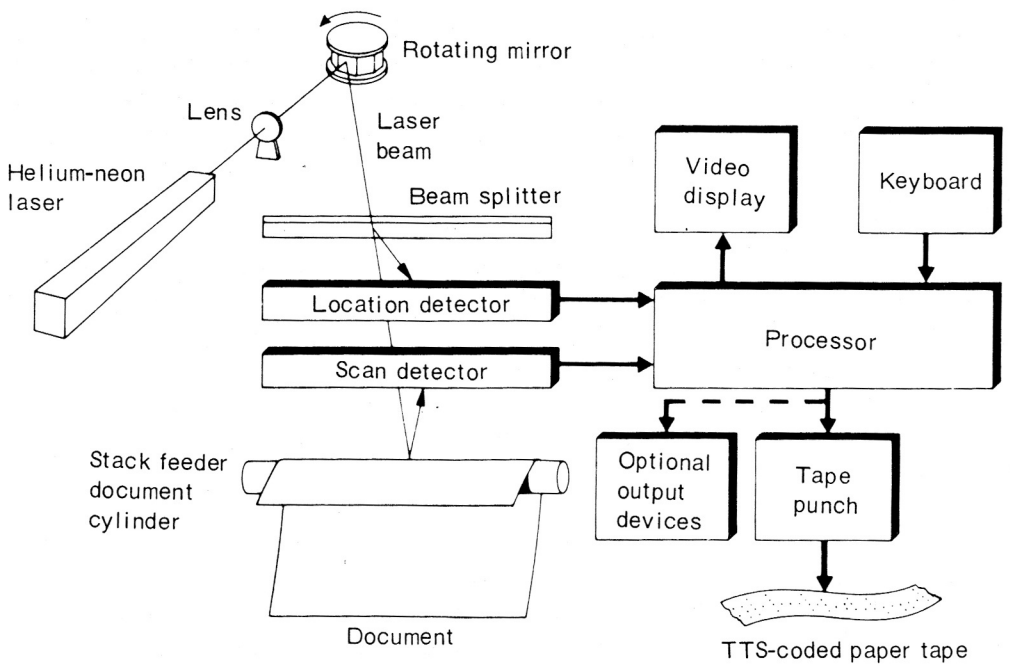
The principle of the flying spot scanner

Another means of achieving the same end without the cathode ray tube and its attendant circuitry has been to flood the entire document with light but only to read or sample the reflected light value of one small spot at a time. This can be done by interposing a spinning disk between the document itself and the photo-

cell which measures the intensity of the reflected light. The disk is perforated with a series of small holes. When these holes are arranged in a spiral pattern it is called a *Nipkow* disk. As the disk spins in front of the photocell, light from different points on the document being read passes through the holes in the disk. This technique has been widely used in the past. It is cheaper, if slower, than the flying spot scanner. However, it has been largely supplanted by the use of laser and multipoint scanners.

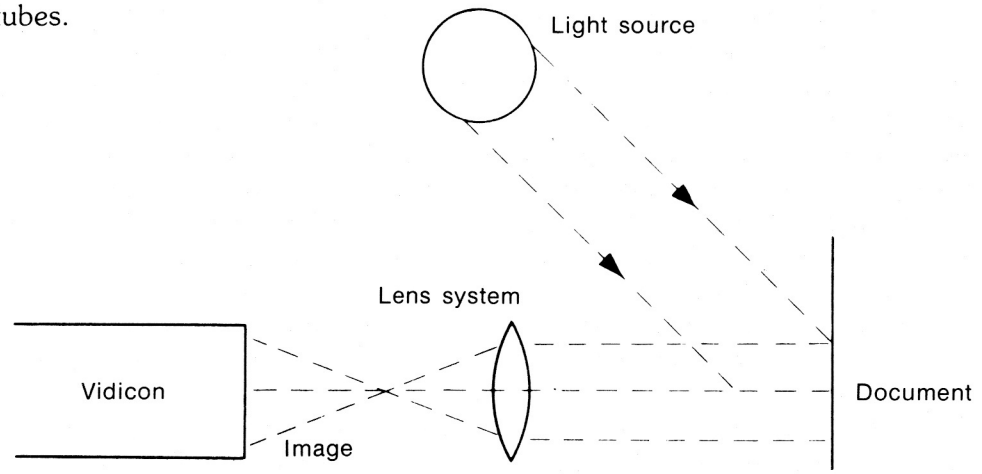
Laser scanners.—A variation of the flying spot scan is the use of a bright beam of light, deflected by a mirror system, to scan the copy being read. A laser is a logical choice as the light source in this scheme since it is exceptionally bright and also because it is coherent light. The latter property means that the focus of the scanning beam will remain constant regardless of variations in distance from the light source to the paper being scanned. A laser beam is deflected across the line being read by a rotating or vibrating mirror. As the light passes across the page, a photocell measures its value as reflected at various points. Since a great many closely-spaced readings are required in order to provide sufficient recognition information, the laser beam will have to sweep across each line of copy forty or more times to generate forty or more slices of data which will be assembled by the recognition logic to build a digital picture of the line which has been scanned.

Because it is deflected into an arc on a flat piece of paper, the laser presents some of the same problems of geometric distortion as does the CRT in a flying spot scanner (or a rotating mirror in a second-generation typesetter). The ECRM 5000 series machines compensate for distortion by means of electronic circuitry. The Cognitronics scanning mechanism splits the scanning beam in two and projects one portion of the beam against a fine-line grating and thus measures the position of the scanning beam with respect to the line being read.



Laser scanning system

Vidicon scanners.—Yet another approach has been to project the image to be scanned onto a scanning tube, specifically a vidicon tube. In this case, the beam of the vidicon tube scans out the patterns of black and white, which are then transmitted to the recognition logic. However, a single vidicon tube does not have enough resolution to scan an entire line of typewritten copy. So, to eliminate the mechanical motion that would otherwise be required to move the scanning tube past the paper or move the paper past the scanning tube, ECRM used six vidicon tubes in its earlier models (700, 1200 and 1800 series). The area covered by the tube overlapped somewhat so that no character image was split between two tubes.



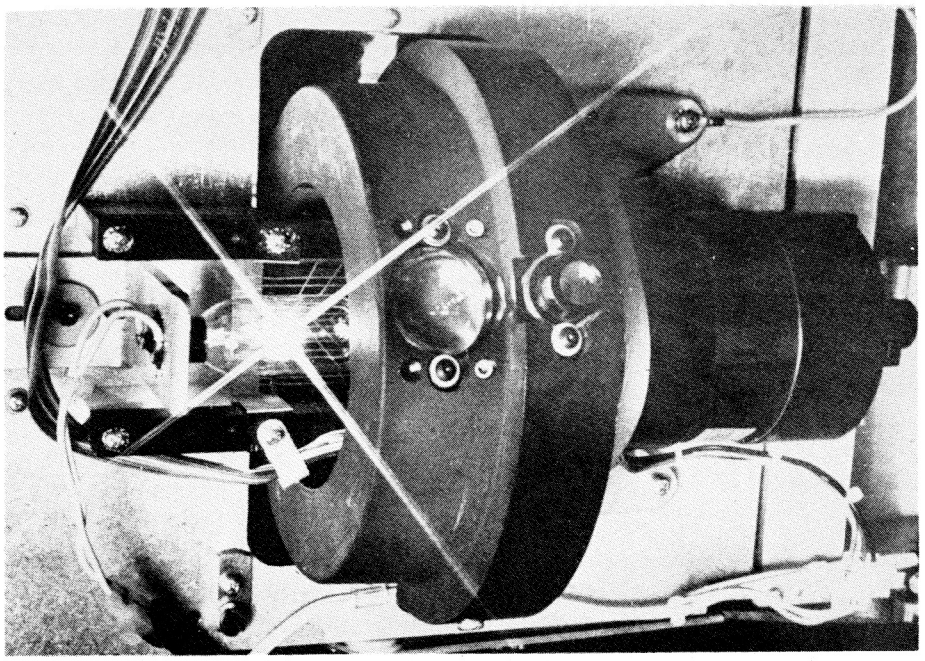
Vidicon scanner

Multi-point scanners.—Of course, rather than to deflect a beam of light across a line to be read, we could take a very small photocell and pass that across the line instead (or pass the line in front of the photocell). In fact, it is now easier and more efficient to move an array of photocells, or to move the paper, line by line, in front of such an array, and to sample a whole series of points in parallel. This eliminates the need for the multiple sweeps required, for example, by a laser scanner. Paper can be moved past the photocells as in the CompuScan 170 and the Hendrix OCR 1 or 2, or the photocells can be moved past the paper, as in the CompuScan Alpha. Originally, the size and weight of photocells posed a problem. Now, however, small, compact photodiode arrays make the job much simpler.

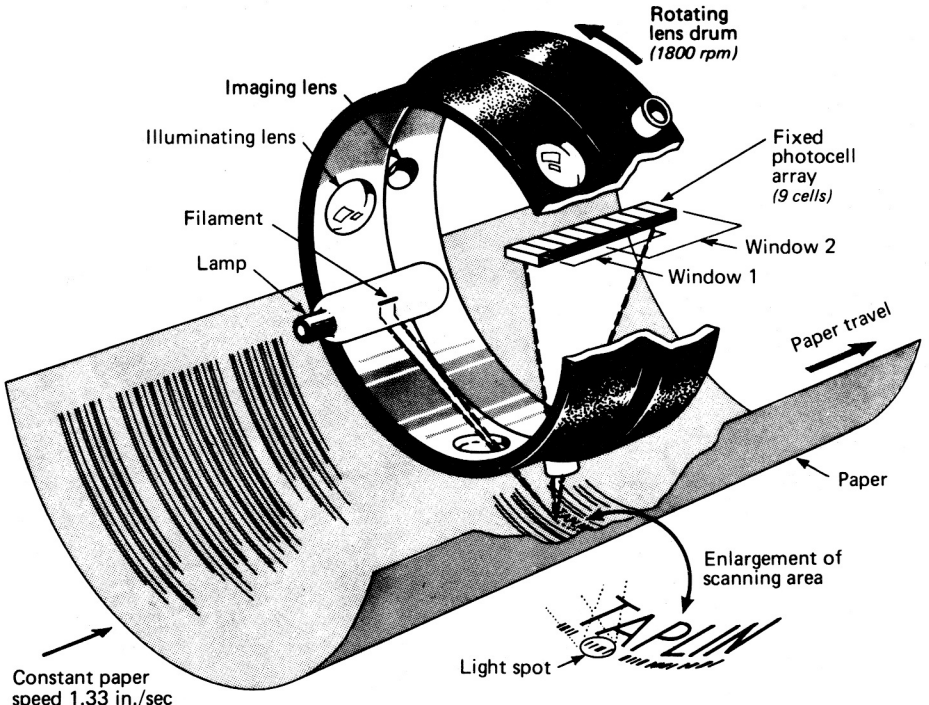
Recognition logic

Once the black and white patterns that make up a line of type have been "captured," it is the task of the recognition logic to identify the characters. Reflected light levels picked up by the scanner are converted from analog (intensity) signals to digital patterns before being submitted to the recognition logic. This is done by matching the light values read against a "threshold" value: any amount of light above the threshold is read as "white" or blank paper, and any value below the threshold is read as "black" or part of a character image.

The first task is to identify individual lines of data and individual characters within a line. All of the readers with which we are familiar scan one line of data at a time. However, so that the reader can accommodate slight skew and mispositioning of lines on a page, the area scanned is always greater than the actual



The optical drum is the only moving part, aside from the paper feed mechanism, on this Model 101 by Context (formerly Taplin). Simple design eliminates back-and-forth motion of the reading mechanism, instead spins the drum at constant speed. Three large illuminating lenses housed in the drum focus light from the incandescent lamp onto the typed line; three small imaging lenses focus bar code images onto the photocell array.



The drum rotates at 1800 rpm, scanning the light beam across the typed line at 1000 inches per second, equivalent to reading rate of 10,000 characters per second when the line is typed at 10 characters per inch. Transport mechanism shapes paper into semicylinder to provide constant focal length for optical system. Three pairs of illuminating and imaging lenses provide three line-scans per drum revolution, giving three "looks" at every character for minimum reading error. Machine's 9-element photocell array is electrically interconnected to form two independent viewing "windows" that handle character misregistration and line skew.

height of one line. Most recognition techniques therefore require that there be at least one horizontal "row" of data—all blank—above and below each line of typed copy. In other words, a line is defined as being an area containing some black data which is bounded top and bottom by areas of white.

These would be identified as separate lines of copy:

```
Here is one scannable line of typewritten copy.

Here is a second line of typewritten copy.
```

These would be identified as *one* line of copy:

Here is one line of typewritten copy.
Here is another line typed with a skew.

These would be identified as *two* lines of copy:

This is one line of xcopy.
This is another scannable line.

Note that the crossout extends into the area occupied by the second line.

These would be identified as *one* line of copy:

This is one line of xcopy.
This line could not be scanned.

The more room between descenders on one line and ascenders on the next, the more latitude there is for skew, and for character cross-outs, and the like. Even more critical, the more room there is between lines, the more latitude we will have if we wish to make insertions between lines of typed copy. Frequently, insertions are to be made after the page has been removed from the typewriter. It is all too easy when you put a page back into the typewriter to mis-position the paper or to "skew" it so that its base line, if projected, would intersect with that of the original copy.

The amount of space between lines is a function of the design of the type font used, the setting of the line spacing ratchet on the typewriter, and the gear ratios of the ratchet itself. Most scanners require that "single-spaced" copy be typed at one-and-one-half line spacing (four lines to the inch rather than the normal six lines to the inch). When the Perry font is used on the CompuScan readers, single-spaced copy may be typed at five-lines-to-the-inch spacing.

Identifying characters.—Once the logic has decided what constitutes a line, it must determine what constitutes a character within the line. Just as they assume that a line must be bounded top and bottom by white space, so it is assumed that a character must be bounded on either side by white space. To insure that the OCR reader can identify individual characters even if the typewriter which produced the input copy is not perfectly adjusted, OCR fonts are designed with somewhat more white space between characters (larger side bearings) than is the case with most other typewriter fonts. Where a font not specifically designed for scanning has been adapted to this purpose (as, for example, IBM Courier 12), it is typed at a wider-than-normal spacing. (A 12-pitch typing element is mounted on a 10-pitch typewriter.)

This is a sample of IBM Courier 12, 8 pt. type.

We have taken a 12-pitch typing element and

The Courier font. mounted it on a 10-pitch typewriter.

Character recognition.—Once the reader has isolated an individual character by its logic, it must decide what that character is. The logic to do this—the real "recognition" logic—is closely and jealously guarded by the various manufacturers. However, most of them use some combination of the following techniques:

- **Matrix matching.**—The digital matrix which describes the character as seen by the scanner is matched against a series of matrices representing the character repertoire of the font being read until the closest match is found. This has now evolved into a reasonably complicated set of procedures whereby character stroke widths are "normalized" to pre-determined values before matches are attempted. Moreover, this technique may be combined with some sort of "feature analysis"—with the latter being used to divide characters into "classes" and fairly rough matrix matches being applied to help identify characters within each class.

- **Feature analysis.**—Characters are classified into categories by *stroke analysis, geometric feature analysis* and *abstract feature analysis.* Geometric analysis would look at attributes such as corners, junctions and crossovers. Abstract analysis would look at density of blackness within the white space, the selection of short characters alone (lower case *x's and z*'s, *s*'s and *a*'s), tall but narrow characters, or tall and wide characters with little mass on the left-hand side, or on the right, such as an *X* or a *C* or a *K*.

Whatever the technique or combination of techniques, most of the logic systems which have been developed make intensive use of table comparisons. In most OCR readers it is therefore possible to change the type font read by changing the recognition tables, although in some cases it may be necessary to modify the recognition logic as well.

Bar code devices

A bar code reader can have much simpler recognition logic than is required by a character reader. It, after all, has only to recognize the bit diagrams created by the specially-created bar code font.



Datatype's DF-2 bar code.

Manual intervention

Frequently, logic recognition will step through a series of "levels" in its attempt to identify a character. One option which has been offered on OCR devices for a number of years has been a mode of operation in which the scanner will stop if it cannot identify a character after going through the first few logic steps, or if it cannot find a match for the character in question within pre-set error limits. The reader will then display for an operator a representation of the character which is causing the difficulty, along with some of the text which precedes it and follows it,

or enable the operator to look into the machine to see where the problem arises by examining the original manuscript. The operator then uses a keyboard attached to the device to signal the identity of the questionable character.

Naturally, an OCR reader will be more accurate (that is, it will mis-read fewer characters) when operating in the human intervention mode than when operating in the normal "best-guess" mode. However, a reader running in this mode will accomplish the input task more slowly, and it will require more operator attention as well. Thus, manufacturers who offer this feature permit its use to be disabled.

Another possibility, being used increasingly, is a kind of "after-the-fact" intervention. The OCR reader does not stop on a questionable character, but flags it. This information is passed on to the next step, and if that step involves the use of a video tube, the text, when displayed on the screen, can show the flagged character (perhaps as a blinking image) so that an operator (who may also be performing editing or composition functions) can determine what the appropriate character should be. Such an extra step on video terminals which would be used for editing the copy anyway is less time consuming than for the OCR device to stop and wait for instructions.

Post-recognition logic

As we mentioned at the beginning of this chapter, the ECRM concept involved the extensive use of what we have called "post-recognition" logic. Generally speaking, this logic performs one of three basic functions: code translating and formatting, editing (*i.e.*, making corrections), and "forms control."

Translation routines.—The simplest form of post-recognition logic is the *code translator*. In its crudest form, this logic simply outputs what has been read, but in whatever code structure the user desires. Quite commonly this is six-level TTS coding, which requires that the routine insert shift and unshift codes into the data stream. Unfortunately, however, the limited signset available on a typewriter is not sufficient to express the variety of typographic symbols and spaces, and the typographic command codes that even relatively simple typesetting applications will require. Hence the code translator rapidly evolves into a *string translator* as well.

A string translator will examine text which has been scanned, and search for the occurrence of certain characters or strings of characters, and replace them with other single characters or with quite complex strings of characters.

A string may "map" several characters into one. In most of the readers sold to this industry, a "1/2" on the typewritten input will appear as a single character "½" on the output, and a "1/4" will be treated in the same manner. Moreover, since typewriters do not differentiate between opening and closing quotes, the scanning logic will take on this task. A double quote preceded by a space would be output as two opening quotation marks. A single quote preceded by a space would be output as a single opening quotation mark. A single quote preceded and

followed by characters would be output as an apostrophe, and a double quote followed by a period or a space would be output as a closing double quotation mark.

A string translator may also convert one character into several characters. However, given the limited signset of the typewriter, it will be more likely to translate several manuscript characters into one typesetting character. Here are examples of some of the more standard conventions:

- Two hyphens will translate into an em dash.
- One to four spaces will translate into an interword space.
- Five or more spaces at the beginning of a line will start a new paragraph.
- More than four spaces in the middle of a line will signal the end of the line.
- *EM or /// may stand for an em space.
- *EN or // will stand for an en space.
- *TS or /+ will represent a thin space.

Such codes, and many others, can be redefined to meet user requirements, for example, ECRM also provides four "Autostrings" and eight "Autofunctions" for which the user can specify his own output coding by means of a "header sheet" or even by means of commands typed directly onto an input job:

Autostrings:
- △ ST = start take.
- △ ET = end take.
- △ PS = start paragraph.
- △ NA = new ad.

Autofunctions:
- △ SL = slug line.
- △ F1 through F4 = call for four type faces.
- △ BF = bold face.
- △ LF = light face.
- △ IF = indent the first line of a paragraph.

The accompanying examples show some of the standard string translations used by suppliers of OCR readers for graphic arts applications. But nowadays it is common for the manufacturer to be able to accept "header" sheets for classes of jobs, or even for individual jobs. A user can define his own output translate table and coding structure—ASCII, EBCDIC, TTS or whatever he might like. He can also, within the memory available within his scanner, make up his own string translations for typesetting formats. One user even applies the string routines to insert discretionary hyphens in words that he knows would be hyphenated incorrectly and to insert "minus value" commands between letter combinations that should be kerned. With enough memory, one can also use string substitution routines to "explode" keyboard shorthand routines, to make corrections to copy by "global" search and replace functions (for example, change every occurrence of "GPO" to "Government Printing Office") or even to do automatic proofreading for commonly misspelled words.

	CompuScan	ECRM	Cognitronics		CompuScan	ECRM	Cognitronics
Em space	///	*EM	ϒms	*Elevate*	/e	*EL	ϒel
En space	//	*EN	ϒns	*Add thin*		*AT	ϒat
Thin space	/+	*TS	ϒts	*Return*		*RT	ϒre
Em leader	/··	*ML	ϒml	*Paper feed*		*PF	
En leader	/·	*NL	ϒnl	*Bell*		*BL	ϒbe
Quad left	/|	*QL	ϒql	*Force upshift*	/Ss	*SH	
Quad right	/r	*QR	ϒqr	*Force*			
Quad center	/c	*QC	ϒqc	*downshift*	/Uu	*UN	
Lower rail	}	*LR	ϒlr	*Vertical rule*		*VR	
Upper rail	{	*UR	ϒur	*End p'graph*	=		ϒep

These represent typical implementations of means of accessing basic typographic command codes. The fact that entries are left blank on this table is of no consequence. These codes can be redefined to meet user requirements, and new definitions can be added. The table is presented for illustrative purposes only. The "Cognitronics" product was formerly marketed by MGD Information Systems, and by Graphic Systems, Inc.

The purpose of translate routines

Why does a scanner for typesetting input need translate routines? We have implied several answers to this question in the foregoing discussion, but perhaps we should state them specifically:

1. Translate routines compensate for the limited character set available on a typewriter to make it possible to express such things as fractions, fixed white space codes, and typographic command functions.

2. Translate routines adapt the scanner output language to the requirements of a particular user's computer typesetting program.

3. Translate routines make input as simple as possible by reducing the need for typographic command codes. The beginning of a paragraph can be signalled by an indent of five or more spaces. Single column text can be signalled by "*t." A 24-point head can be signalled by "*24." This makes it possible for editorial and classified ad personnel to prepare input for typesetting programs without having to learn the typographic commands required by the program. It is doubtful that nearly as many OCR readers would have been sold to newspapers without this function.

Of course, one does not need an OCR reader to perform string translations and substitutions. One could perform the same functions in the computers of the typesetting system, or even in the front-end controller of a phototypesetter. However, most customers for this type of equipment do not have an in-house programming staff for typesetting software. The OCR readers perform these functions because they *must* perform them if they are to be sold as stand-alone units to feed into existing typesetting systems. When scanners are sold as part of

a total "system," it is no longer of any consequence where the functions are performed. Thus, the Hendrix OCR 1 and 2 and the ECRM 4400, when sold as part of a system, do not contain any of the post-recognition logic we are presently discussing. Most post-recognition functions are performed by the computer software in the *video terminal* system.

Beyond translate routines; formatting for fielded data

These string translation programs are intended to be used for input of text copy and are available in one form or another from all of the manufacturers actively courting the graphic arts market. One manufacturer, CompuScan, offers a different type of input format program in addition.

CompuScan calls its program "Freeform III" to indicate, we suppose, that fielded data may be entered in a "freeform" fashion without having to observe fixed or arbitrary field lengths. The freeform language is, for all practical purposes a high-level compiler in which the user can write a "program" to format his data. This program can output the information read in either fixed-field or variable-length field format and it can be made to manipulate or to validate the information in any of the input fields. The program can verify, for example, that field "three" of an input record contains only numeric information with a value of between 5 and 99, but not 45. The same sort of validation can be done equally well by subsequent computer program, of course, but the scanner has the advantage of being able to reject the document in question and to sort it into a "reject" bin so that it can be corrected and re-input (provided, of course, you can figure out why the scanner rejected the input.)

The program is intended for a "data processing" type of input. And, indeed, it borrows heavily on established techniques for this type of input. But there are many applications in a typesetting environment in which some or all of the information to be input will be processed by computer data manipulation programs. Classified ad forms come readily to mind in the newspaper industry. In the commercial field, dictionaries, product guides, telephone directories, price lists, schedules and time tables, book catalogs—almost anything which consists of "entries" which appear in a consistent format—could benefit from a program such as this.

Editing or correction routines

The original thrust behind the current use of OCR readers in the newspaper industry, in particular, was that reporters and classified ad takers could create their own input in OCR-readable form, correct their own errors or add second thoughts about wording and punctuation, then pass the typewritten copy on to someone else who may make further changes before the manuscript is finally fed into the OCR reader.

This implies that the OCR reader has "post-recognition logic" capable of coping with corrections which have been properly indicated on the page to be

scanned. Several techniques were developed for this purpose and embodied in the post-recognition logic of the original ECRM Autoreaders. Some of these techniques had been used previously, but many others were unique to ECRM. Other suppliers have followed this pattern, although with variations, and have even added others.

Delete Symbols

Frequently a typist knows or feels that he or she has made an error. Or a typist may change his or her mind after typing a word or more. For this reason, the OCR readers for typesetting input all permit the use of "delete" symbols.

```
Delete the preceding charr¤acter.

Delete the preceding wrd¤¤word.

Delete the whole drn sntence¤¤¤

Delete the sntnc¤¤¤Delete the sentence up to here.
```

CompuScan takes an approach (above) which has been followed for a number of years. One character out of the typewriter's character repertoire is chosen as a delete symbol. (This is generally a lozenge on the Perry font used by CompuScan.) One delete symbol tells the scanner to delete the preceding character. Two delete symbols tell the scanner to delete the preceding word. Three delete symbols tell it to delete everything which has occurred on the line up to that point.

```
I ti#yped the wrong cah##haracter.

Now I want to delete there###ree characters.
```

ECRM, on the other hand, generally uses separate symbols for character and word deletes. If the sign # will delete the preceding character, ## will delete the preceding two characters, ### will delete the preceding three characters, and so forth.

```
I want to dill+ delete this word, nnd niw++ and now
two words.
```

The ECRM word delete symbol works in the same fashion. If the sign + is chosen as the word delete symbol, + will delete the preceding word, ++ will delete the preceding two words, and so forth.

Delete symbols.—When a typist senses or "feels" the making of a typing error, or changes his or her mind after typing a character or a word or more, delete symbols can be used. CompuScan pre-empts one character out of the typewriter's character repertoire as a delete symbol. One delete tells the scanner to ignore the preceding character. Two symbols tell the scanner to ignore the preceding word, and three symbols ask it to ignore the entire line, up to that point only. ECRM differentiates between word delete and character delete symbols. There are other suppliers who use the CompuScan approach.

Clearly the use of delete symbols can make it possible for even an inexpert typist to prepare copy for OCR input. The performance of a good typist is also improved, since little time is lost in correcting errors. However, an extensive scattering of such symbols makes the manuscript more difficult for a subsequent reviewer to follow.

Strike-outs.—The delete symbol is not helpful in every case. It can be used only if you catch your error or change your mind at the time you make the error. Once you have typed well beyond that point it is too late. Here the cross-out is more effective. A strike-out or cross-out is a procedure by which a person draws a line through the letters or words he wishes to delete, thereby making it impossible for the scanner to recognize the character or word. The cross-outs must be made carefully, usually with a nylon-tipped, dark-colored pen. It may be performed either by the person who creates the copy or by someone who reviews it later.

- **Vertical cross-outs.**—A single character deletion must be a vertical line. It must be made carefully. ECRM suggests that the delete mark start in the middle of the character and extend upwards. MGD insisted that the vertical delete extend to the full height of any possible character.

This cross-out would be acceptable to an MGD device but not to ECRM:



This would be acceptable to ECRM, but not to MGD:



This cross-out would not be acceptable to any scanner because it intrudes into the space occupied by the following character:



- **Horizontal cross-outs.**—A deletion of two or more characters may be made by means of a horizontal cross-out. Some devices assume that an entire word must be so deleted. Others permit the elimination of a portion of a word as well. However, since the logic does not automatically insert a space when you make a deletion, you must be careful not to delete the spaces on either side—or both sides—of a correction.

- **Multi-line deletes.**—MGD scanners offered a feature which other manufacturers do not provide—a multi-line delete. If you wished to delete an entire paragraph you could of course simply cross out every line, but with MGD's approach if you crossed out the first line and drew a vertical line beside the rest of the copy to be deleted, the program would omit the entire portion.

Insertions.—It is also necessary to cope with insertions, usually added by the reviewer or the editor. Most of the OCR readers have been programmed to hold two lines of copy at one time in their buffer memories. Whenever the logic encounters a horizontal or a vertical cross-out it will jump to the end of the line on which the cross-out appears and search the next line for an insertion. If it encounters an insertion it will replace the copy crossed out with the insert copy. If it does not, it will simply delete the crossed-out copy. To enable the scanner to identify insert copy, a delimiter code must be placed before and after the insert. If there is more than one replacement on the line, the software will process them in sequence. If there is a deletion on the line followed by a replacement, the typist will have to type a "dummy" insert so that the logic will not apply the replacement at the wrong place.

MGD and Graphic Systems {Bobst}: This is an ~~xxx~~ delimiter.
+ insert +

ECRM: This is an ~~xxx~~ delimiter.
/insert/

CompuScan: This is an ~~xxx~~ delimiter.
Δ insert Δ

ECRM Here we ~~are going to~~ make ~~deletions~~ and ~~substitutions.~~
// /substitutions/ /deletions./

CompuScan: Here ~~xxx~~ several ~~wrds~~ to ~~xx~~ replaced.
Δare Δwords Δbe Δ

Sample insertions from MGD, ECRM and CompuScan.

ECRM Here we forgot to|a word.
/ add /

- **The limitations of inter-line insertions.**—If we are to make inter-line insertions and replacements, the original copy must offer space to do so. This means that copy which is to be so corrected must be typed at twice normal spacing (two lines to the inch rather than four) to allow room for inserts and replacements to be added between the lines. The scanner, however, must be instructed to look for insertions. Naturally, to run the scanner in this mode slows up the reading process. More important, this method of handling corrections requires that the person making the corrections, put the page back into the typewriter in order to type the corrections, and it must be put in carefully to avoid skew.

  This led to a division of labor in some newspapers. Editors might mark corrections in ink which the scanner could not "see," and typists or composing room personnel (depending upon the labor contracts prevailing) would actually make the cross-outs with a dark-colored pen and type in the replacements and insertions.

  It should be clear from the examples we have shown that these correction facilities are useful only for relatively straightforward and relatively light copy editing. There is insufficient space available to add more than a short phrase, and there is no way to transpose or rearrange blocks of copy. Several of the manufacturers have devised ways of combatting these limitations.

- **Cut and paste.**—One of the concepts embodied in the ECRM scanners is that editors can handle major revisions by cutting copy and scotch-taping it back together again. This is one of the reasons why the ECRM scanners will accept input sheets of virtually any length and why they are quite tolerant about variations in paper thickness. However, even so, the "paste up" must be done carefully.

- **Paragraph insertion.**—CompuScan offers a "paragraph insertion" program for its scanners. This is a means of inserting more copy than can be typed between the lines on original input. The standard program will accept one or more inserts of up to 300 characters in total length. An expanded program will accept more inserts or larger inserts. The copy to be inserted must be encountered before it is to be used. Generally this means that it is typed on a separate sheet of paper inserted at the beginning of the job, or, if there are a number of inserts to be made, just before the page on which the insert is to appear. The text of the insert is stored in the memory of the scanner, then called out to be inserted when the appropriate code is encountered.

Forms control

Some manufacturers offer a kind of logic which might, in fact, be called "pre-recognition" logic since it determines what material on a page is to be read, and in some cases even the sequence in which it is read. This is what OCR device manufacturers call "forms control." The simplest type of forms con-

trol is the logic which determines when the scanner has encountered the end of the typed copy on a page. Generally speaking, if it encounters more than a specified number of blank lines, the scanner will assume that it has read all of the data on the page. A more complex program can be used for classified ad input. This logic might tell the reader to skip the first four inches of a page (where billing information might be contained) before it starts to read data. Or it might tell the reader to scan the page out of sequence. More complex still is logic which instructs the OCR reader to concern itself only with certain areas on a page and to skip others. A header sheet might convey this information to the scanner and also tell it what codes to input before and after the specific fields to be read.

Type fonts for OCR input

The scanners we are discussing in this chapter have been designed to read only one font, or, at best, several fonts, all of which were intended to be scanned. On the next page we show samples of the fonts generally used and indicate which fonts are most popular with which manufacturers. Even the most stylized font (OCR-A) is quite acceptable to editors and general readers once they become accustomed to it.

The preparation of material and maintenance of equipment

It must not be assumed that "any old" typewriter can prepare input which is acceptable to a scanner. The device must be maintained with great care, and often typewriter repair persons unfamiliar with the requirements of OCR are not capable of providing this level of maintenance. Since IBM typewriters are used almost universally (at least in the United States and Canada), IBM has begun to develop special cadres of mechanics who are appreciative of the subtleties and the standards. Some large OCR users have trained their own personnel. Carbon ribbons are, of course, essential. The quality of paper is also extremely important. Moreover, some degree of training—not only in the conventions for insertions and deletions, but also in the spacing of material on the page—is required. Nevertheless, despite these obstacles, OCR offers a viable method of input and its principal attraction is the opportunity to capture keystrokes at their source.

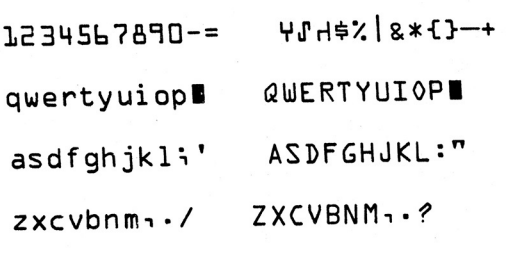
Shifting responsibility to the writer

The capture of keystrokes does imply that the writer is to be held accountable for the accuracy of his input, including grammar and spelling. While editorial changes can be made in the fashion we have described, if they are substantial the entire process breaks down. Some writers are unwilling to accept this responsibility or are unqualified to perform under it. (They might be non-spellers, for example.) On the other hand, many writers feel that they have more assurance that their material will not be mutilated by compositors who insert their own typos and "pi" the lines.

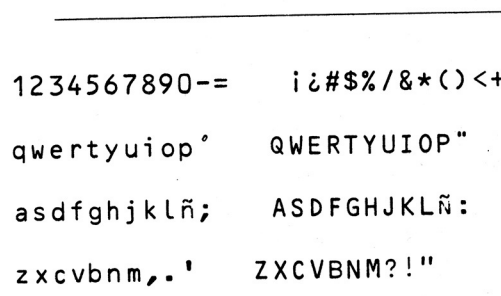
So far as accuracy of input is concerned, when the material has been properly prepared, and the OCR device is in proper adjustment, misreading is less likely

to occur than would be the case with a tape reader. One error in 10,000 characters read is claimed as a maximum by most manufacturers, and some make even more extravagant claims. With the use of supervised reading, whereby the machine stops when it has trouble with the recognition of a character, one error in 25,000 characters is more common. The problem arises when the device is reasonably certain that it has read the correct character rather than when it is confused about the character it is trying to read.

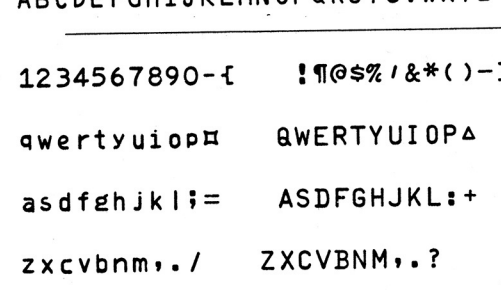
Samples of some commonly used OCR fonts



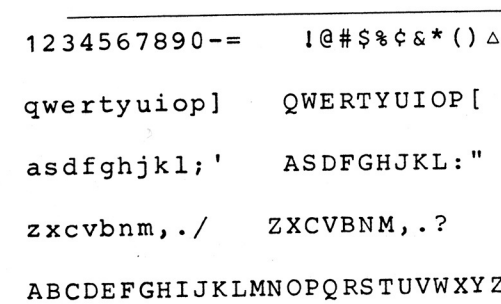
OCR-A *is generally acknowledged as being the easiest of the standard fonts for an OCR reader to recognize. It is also one of the most "stylized" in appearance from the standpoint of human legibility. Because of its large size, OCR-A leaves less white space between lines than do other OCR fonts, which means less tolerance for skew and less latitude for inter-line insertions. It has been widely used in the U.S. outside of the graphic arts industry. It is commonly used on billing forms which are to serve as input into OCR systems. Commercially available numbering machines used for imprinting forms also print OCR-A numerics. For this reason CompuScan offers programs to read OCR-A numerics.*



OCR-B *was created for the European Computer Manufacturers' Association by a type designer in collaboration with engineers, with the intention of creating a more conventional appearance. It was designed to be printable by a wide range of devices, to be aesthetically acceptable, to cause low error rates, and to use the lowest possible print quality. The characters were made as inherently different as possible in the light of pure information theory and were not designed for any particular system. OCR-B has been widely used in Europe, but very little in the U.S., probably because U.S. manufacturers have preferred the U.S.-designed OCR-A, commonly applicable to non-graphic arts OCR readers.*



Another widely used font is the so-called ***Perry*** *font. Originally developed for use with the REI scanners at Perry Publications, the Perry font was adapted by CompuScan for use with its 170 series page readers. Like OCR-B, the Perry font is a* sans-serif *face of moderate size. It is most readily distinguished by the fact that it has no descenders. All characters actually sit on the baseline.*



IBM Courier 12 *is the fourth font which is commonly used in graphic arts applications. This is a* serif *design which is widely used for general correspondence on IBM typewriters. It is a 12-pitch (12 characters to the inch) design, but it is typed at 10-pitch (10 characters to the inch) spacing to insure adequate separation between characters. Courier is used by ECRM. CompuScan and MGD both offer recognition routines which will read the font, but neither advocates its use.*

Summary

It may be seen that the introduction of OCR may imply a restructuring of the graphic arts industry, since it carries with it the potential of capturing the writer's keystrokes, eliminating much of the initial input (at least for certain types of work) and the consequent proofreading. Many users have in fact dispensed with all proofreading, on the assumption, which often holds true, that the reader and the "downstream" processing and output do not create their own errors.

But it is also true that even when manuscripts need to be retyped, OCR sometimes offers a more economical method of generating input, even for quite complex work, such as mathematical formulae. This is to say, the input can be created more economically than would be the case with on-line terminals or tape perforators. This is especially likely to be the case where the input requires a large number of operators (and the investment in alternative input devices would be substantial), and where input is highly seasonal or sporadic.

OCR has also found its way into many "word processing" applications. Consider, for example, the case of a secretary whose duties require so little typing that the exclusive use of her own word processing station would be uneconomical. She could mount a scannable typing element on her typewriter, then pass the pages on to a production pool where subsequent revisons could be effected.

As we shall see, in the graphic arts industry OCR has been challenged by the advent of relatively low cost video terminals. The market for OCR devices in graphic arts will surely not be as great as some analysts had once predicted. While OCR readers have come down significantly in price, and prices may drop still further, the price of video terminals and other means of input may, in the future, decline still more significantly. At this writing, it is difficult to predict what the ultimate role of OCR will be, except to suggest that this technique will certainly not disappear in the near future.

Review Questions:

1. What are some of the factors which lead to the relatively widespread use of optical character recognition devices in the graphic arts industry?
2. What is "omni-font" recognition logic? What might some of its applications be?
3. How does the "ECRM concept" preserve the "paper flow" within a newspaper production environment?
4. What is the difference between on OCR and an OBR device?
5. What are the different components that make up a scanning device?
6. Describe three different scanner technologies.
7. At what point in the scanning procedure does the analog-to-digital conversion take place?
8. Why must copy intended for OCR scanning be typed with wider than normal spacing between lines?
9. Describe the techniques used in OCR devices for actual character recognition.
10. What is meant by "post-recognition logic" and what are some of its applications?
11. What is the difference between a code translator and a string translator?
12. Give examples of types of string translations employed by different OCR devices.

18

Video Terminals and Video Editing Systems

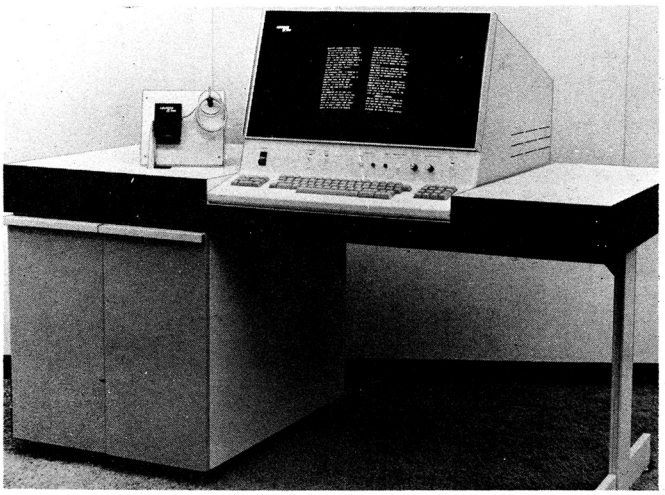
IN THIS CHAPTER we shall begin our discussion of video terminals and their impact upon the graphic arts industry. While such terminals were used in conjunction with computers as early as the 1950s, they made little headway with the graphic arts industry until the 1970s. Today, most of the video terminals that are used in our industry have been specifically designed for text processing and typesetting applications, just as the OCR devices which "caught on" in the industry were also specifically engineered for these purposes.

Ignoring a few early systems that demonstrated the feasibility of applying video terminals to text editing, our story really begins with the introduction of a "stand-alone" terminal developed by the Harris Corporation and first shown at the American Newspaper Publishers' Association Institute exhibit (ANPI/RI) in New Orleans in June 1970. The immediate acceptance of this terminal—called the "1100"—ushered in a new era not only for the newspaper industry but for all practitioners or potential users in all aspects and facets of typesetting and related text processing—both in "production" and in "editorial" environments.

Other manufacturers or suppliers were engaged at the same time in the development of similar terminals. Indeed, one such supplier (Hendrix Electronics) had a product at the same show. But the quality of the image of the Harris 1100, and the general responsiveness of this terminal, made a "hit" at the show. It thus inaugurated a new era, spawning many imitations in the years to come as well as inspiring subsequent improvements, by Harris and others, in functional concept and design.

The 1100 was shown as a "paper tape in-paper tape out" editorial tool. It differed from other "tape perfecting" devices in that it provided an internal buffer or text storage area to minimize the serial nature of the correction process, for, with respect to text displayed on the screen, corrections could be effected randomly. The terminal also offered the new capability of *cursor editing*.

Harris had not initially conceived of the 1100 as a production tool. It had visualized the terminal as one to be used by editors, mostly to make changes in



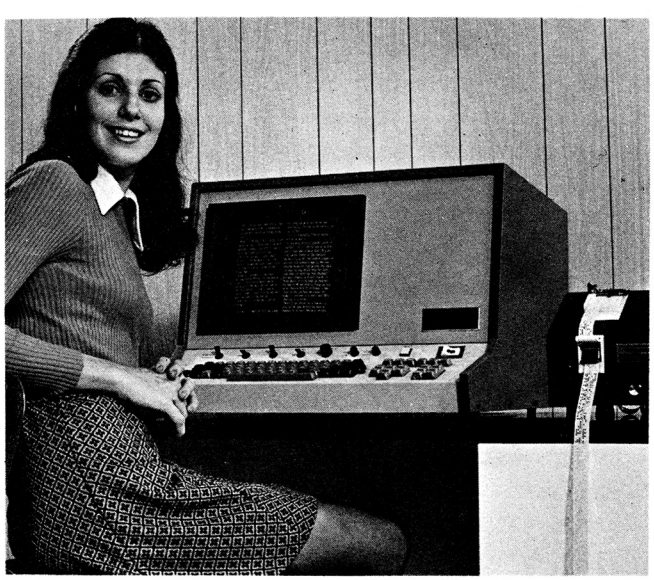
The Harris 1100 terminal, a hard-wired device which used paper tape as its I/O medium, was introduced in 1970. It displayed upper- and lower-case characters on the screen in single- or double-column format.

wire service stories received at the local newspaper level in the form of Teletype printout and paper tape. Selecting his story from an inspection of articles presented by the "hard copy" rolling from the printer, an editor could mount the corresponding paper tape, read in a screenful of text, review it, and enter editorial changes, while instructing the device to punch out a new tape of the revised story. Twenty-five lines could be "read in" and edited at one time, with 80 characters across the screen, or 50 lines, each of 40 characters, could be presented in a two-column format. Hence enough information was available at one time to get the "sense" or "flow" of the story.

But what the industry saw in the 1100 at that time was not primarily a method of editing. It was rather a better method of "tape perfecting" that was faster and easier than any of the existing "pre-computer" (that is, ahead of the computer in the production cycle) paper tape merging devices. It was also more interactive than the batch correcting and merging techniques offered by existing computer composition systems.

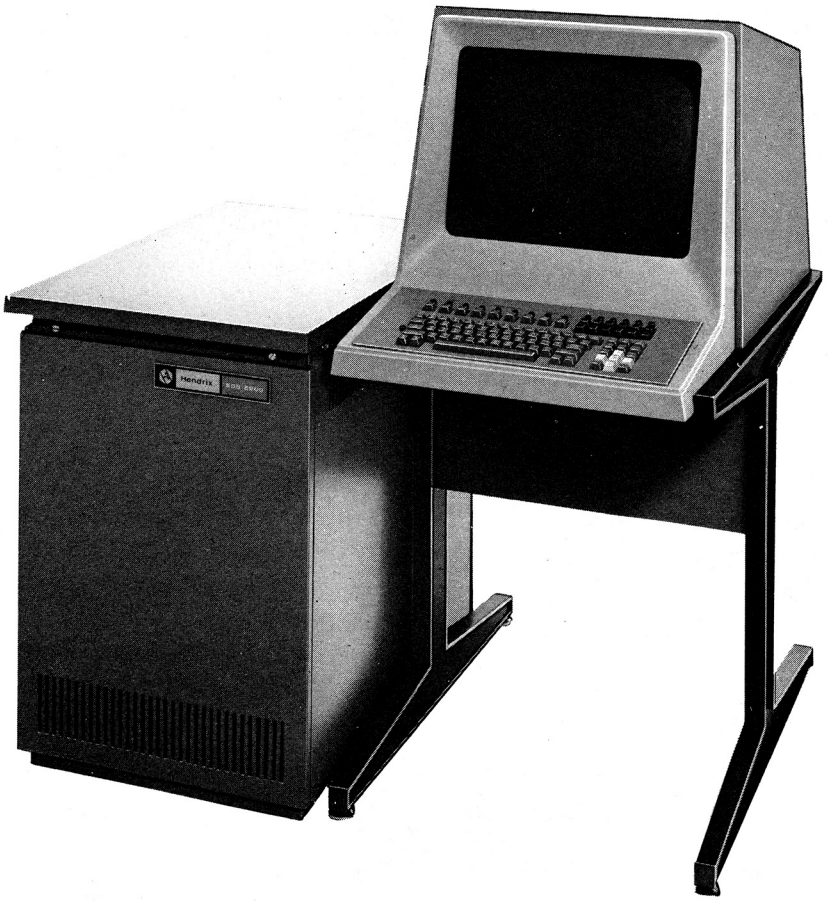
Within a year—or certainly within two years—many similar terminals had come into being. Hendrix withdrew its 5102[1] offering and introduced the 5200. Mergenthaler marketed the CorRecTerm, and Omnitext came out with its 1500, which sold for a while under the "Mohrtext" name, and subsequently as the Singer 8400. All of these were configured to receive their input from paper tape and, in turn, provide paper tape output. They could be used for editing, for proofreading and correcting, or merely for the implementation of corrections already noted from reading printout or preliminary typeset copy.

1. Hendrix Electronics developed an editing terminal in 1969. A prototype of the terminal, which was called the 5102FD, was installed in the office of the Associated Press in New York City in 1969. Several other models were installed in various sites.



When used directly for proofreading, the purpose of the activity is not only to verify that the text has been keyed correctly, but, if not, to make necessary changes at the same time that errors are detected. When used for "correction implementation," the purpose is to locate and eliminate those errors which have already been marked on printout or galley.

Left: Mergenthaler's CorRecTerm (note the emphasis on CRT) was also a hard-wired terminal which was released in 1971. It, too, offered single- and double-column display modes, and was a paper tape-oriented device.



Left: The Hendrix 5200 terminal, a redesigned version of the 5102, was presented for the first time at the ANPA conference in Milwaukee in January 1971. It, too, used hard-wired logic and could perform the basic editing and correction functions found on the other contemporary terminals, *e.g.*, overstrike, insert and delete. The 5200 operated either with paper tape input and output, or on-line to a computer system.

Below: The Omnitext 1500 terminal, announced in 1971, differed from some of the other early graphic arts terminals in that it was built around a minicomputer rather than a hard-wired controller.

Basic components and functions

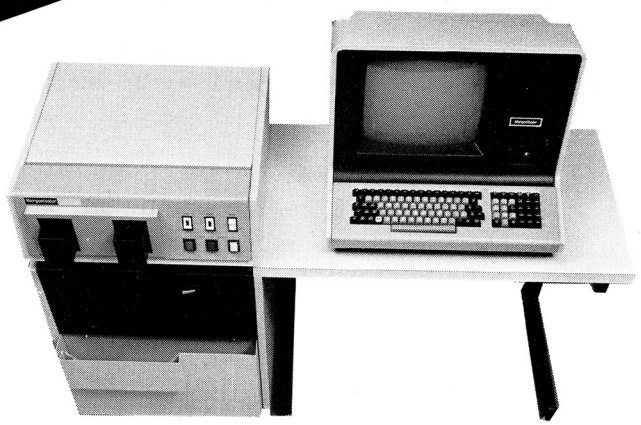
We have made use of the name "terminal," although a stand-alone device, such as these units started out to be, is not truly a "terminus" or adjunct to a "system" in any on-line sense. It *is* an adjunct to a system, of course, in a production sense. But because the VDT (video display terminal) resembled computer video terminals such as those found at airline counters for communicating with large computer network systems, the name "terminal"—at least in the graphic arts industry—has been associated with these "stand-alone" devices. They are often called VDT's (for video—or visual—display terminals), but the British have more correctly labelled them as "VDU's," for "video display units."



The terminal provides a keyboard and a video screen. On the keyboard will be found—alongside the usual typewriter-like arrangement

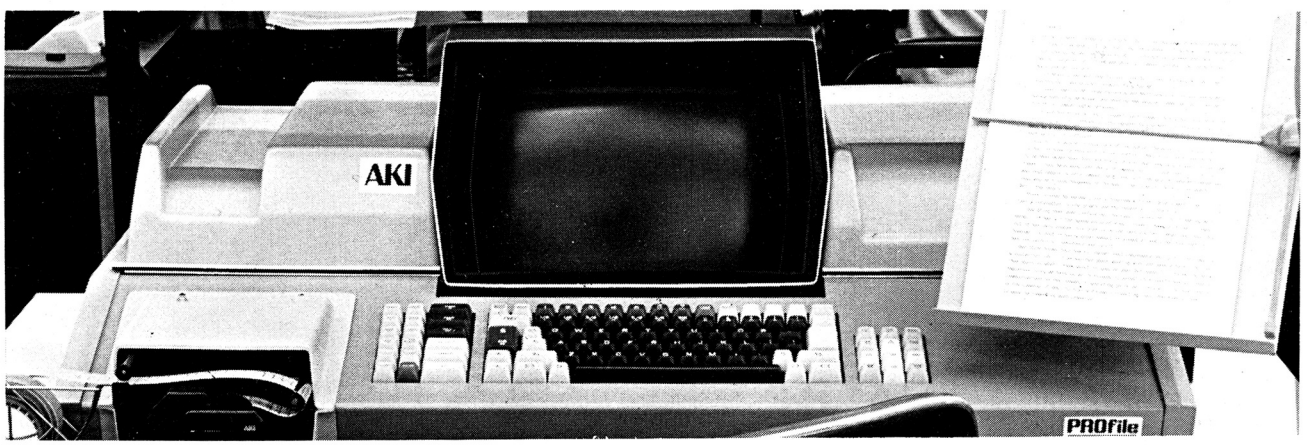


Above: Automix Keyboards, Inc. (AKI) introduced the Ultra series line of editing terminals in 1972. Instead of using hard-wired logic or a minicomputer controller, each terminal had its own built-in microcomputer (a "computer in a chip"). The AKI terminals made use of either paper tape or magnetic cassettes for text storage and revision. In place of the usual cursor control keys, a track ball is used to move the cursor about on the screen. A plexiglass screen, mounted over the CRT display, generates troublesome reflections.

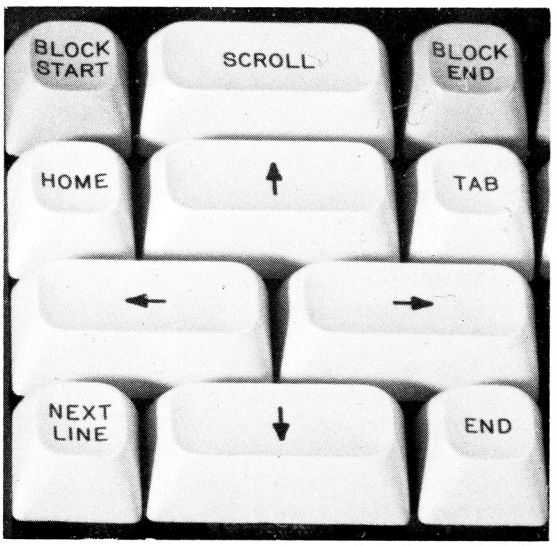


Above: Mergenthaler's MVP editing system, launched in 1976, also uses a microprocessor controller. It is housed in a separate box or cabinet which contains a paper tape reader/punch as well. Floppy disc drives are an option available on this single terminal "system," as is an interactive interface which allows the terminal to use the h&j logic of Mergenthaler's V-I-P phototypesetter, and to review those line-ending decisions on the display.

Below: AKI's PRO series terminal, introduced in 1974, made use of the same built-in microprocessor. The screen is low, but well-angled towards the operator. It is nicely hooded and has an effective anti-glare treatment (no plexiglass screen in front of the tube). The terminal displays 14 lines of 64 characters each. Characters can be displayed in a "bold" or an underlined mode, as well as the normal mode. The track ball has been eliminated.



of key buttons—a set of *cursor control* keys. These are used to direct a "cursor" or pointer to move from line to line and from character to character on the screen proper, thus signalling a specific location—an individual character—within the text displayed. A position is thus identified where an *operation* may take place, such as to "overstrike" and substitute another character for the one occupying that particular location.



Typical grouping of cursor control keys. This particular key cluster forms part of the keyboard layout on the Omnitext 1500 terminal.

As we have mentioned, the screen shows a display of characters across its face, either in a single or double column format. The character "set"—unlike that of most "computer" video terminals—comprises both upper- and lower-case characters, and also presents certain common typesetting symbols. The characters are "refreshed" so that they remain on display in a relatively flicker-free state until they are changed by keyboard command or the unit is turned off.[1] The characters on the screen are written into the buffer of the terminal from a paper tape reader, or by direct keyboarding from the unit itself.

1. See discussion of "refreshing" on page 293.

The buffer is usually organized in what is called "efficient mode"—which means that memory positions are assigned only where actual text is written, and not where end-of-line blanks are to appear. *Display screen mode* would require a position for the blanks as well.

The first VDT's made available to the industry provided only a "screen-sized" buffer, although subsequently this buffer was optionally enlarged. By "screen-sized," we mean that the buffer contains only that number of characters which are displayable on the screen at one time. If the unit is designed, for example, to display 25 lines of 80 characters each, the buffer would consist of 2,000 character locations. But if a 4,000-character buffer is provided with a displayable area for only 2,000 characters, more room is available to edit text randomly, and, as we shall see, it is then possible to scroll text back and forth across the "window" of the tube.

Let us assume, then, that the text which flows into the buffer is stored serially, in the "efficient" mode. How are character images generated on the tube?

This is accomplished by the use of integrated circuit chips, known as ROMs, or read-only memories. Each character identity code is communicated to the ROMs and translated into a dot pattern[2] (Where programmable character sets are offered, RAM replaces ROM, so that the user can define his own character images and store them in random access memory.)

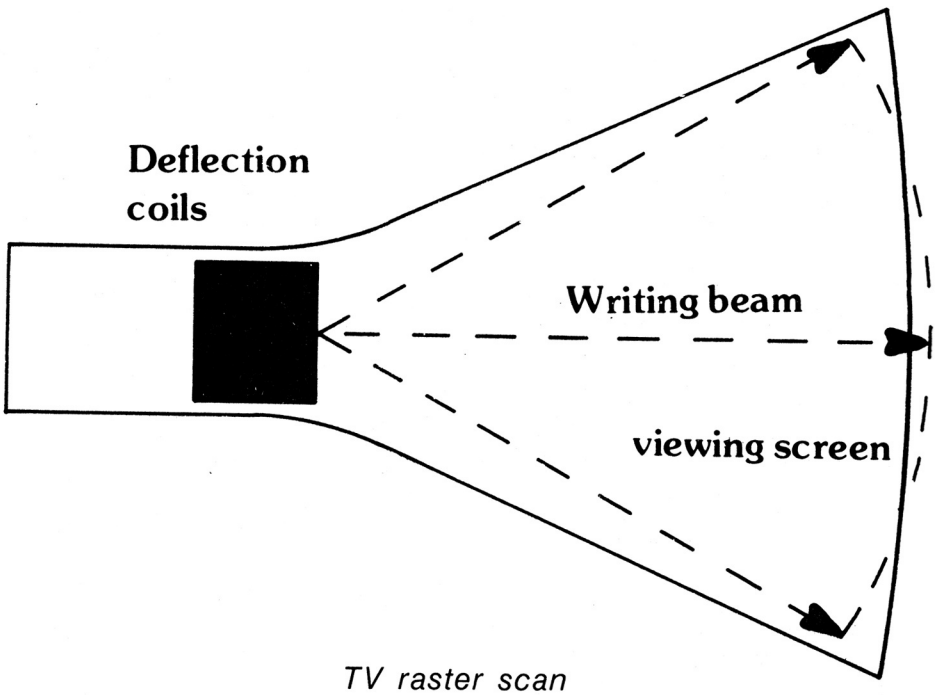
2. We are describing a "TV"-type video display. There are a few such terminals which use "storage tube" technology so that the screen does not require refreshing. There are, however, certain liabilities associated with this technology. There are also some editing VDT's which use a calligraphic or random-stroking character-generating technique.

Most characters are created by the use of dot matrix pattens, although a different kind of tube (which strokes characters or constructs them from vectors) could be used. The TV tube is most common, and its characteristic is that it functions in a "raster scan" mode. The electronic beam moves horizontally from left

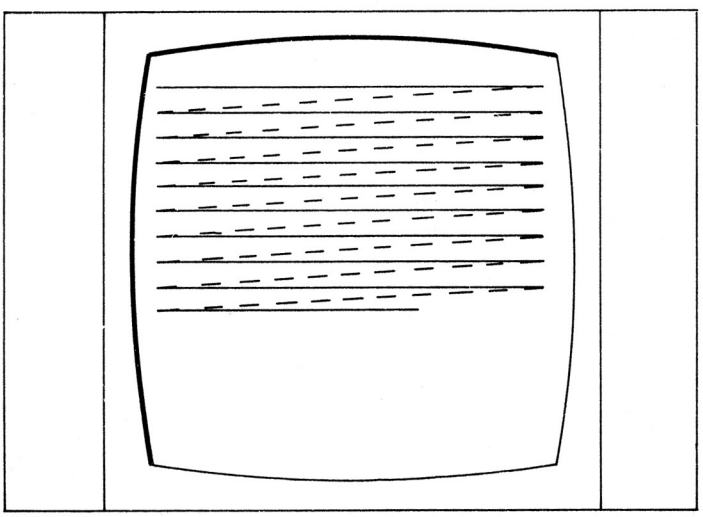
Cathode ray tube

Cross section of a cathode ray tube as viewed from the top. Note that the writing beam is deflected in an arc. The face plate of the tube is somewhat curved, but it would be very difficult to read if it were curved into the full arc described by the beam. If no corrections were applied, the image on the face of the tube would appear fuzzy and out of focus in the center of the tube.

Therefore, both the focus of the writing beam and the rate of deflection must be adjusted by correction circuitry as the beam is swept across the face of the tube.



TV raster scan



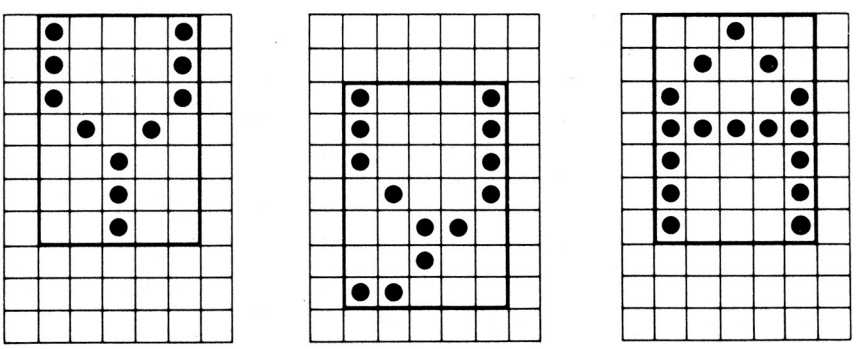
American broadcasting television uses 525 horizontal lines across the tube. The number of scan lines is constant, regardless of the size of the tube. Thus, the scan lines of a 24″ tube are spaced twice as far apart as those on a 12″ tube.

European TV uses 600 scan lines, which is one of the reasons European TV pictures generally look better than those on American TV, and some manufacturers of video monitoring devices opt for European-compatible TV displays in order to get their higher resolution.

For video terminal applications, the tube may be mounted either vertically or horizontally and the deflection yoke assembly can be rotated 90° with respect to the tube. It is therefore possible to have a horizontally mounted tube with vertical scan lines, or a vertically mounted tube with horizontal scan lines.

Character matrices

Upper- and lower-case Y's and upper-case A formed with a 5x7 dot matrix. Total area occupied on the screen by each character is actually 7 positions wide by 10 deep, but character design must be confined to the 5x7 area indicated by the heavy box. Note lower-case character displaced two positions below base line.



to right, writing a series of dots wherever required by the character matrices. Then the beam is shut off and re-directed to the left of the screen, where it creates the next scan line of dots. This normally happens at the rate of 60 cycles per second, so that the entire image of each character is refreshed 60 times per second. There are a few video terminals where the raster scan occurs only on alternate scan lines each 1/60 of a second, which means that the first, third and fifth lines would be refreshed (*etc.*) before the second, fourth and sixth horizontal lines are. This is known as "interlacing" and it is a method which fools the eye into thinking that the character image is more persistent than it really is.

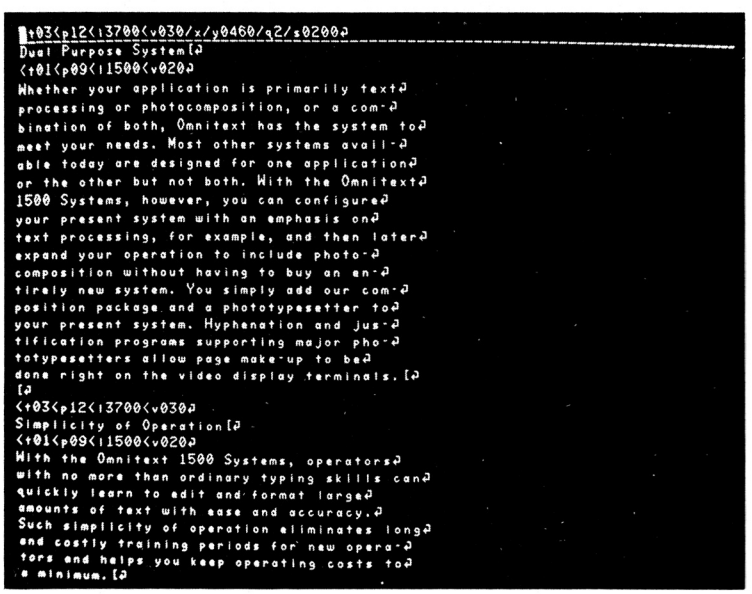
The dot matrix out of which each character is formed usually consists of a monospaced area of, say, seven dots horizontally, and perhaps nine lines vertically, with perhaps two blank scan lines between text lines. At least a 7 × 9 dot matrix is necessary to display adequately upper and lower case characters, although it is also possible to "drop" the entire pattern of a 5 × 7 matrix by one or two horizontal lines where there are characters with descenders to create the effect of a 7 × 9 matrix, or to perform other comparable manipulations.

And so we have, then, a keyboard with cursor controls capable of identifying a location within the buffer, and perhaps of "scrolling" lines up and down across the "window" of the screen to the extent permitted by the size of the buffer. What are the cursor editing functions which can be performed?

Cursor editing

The most common method of operating a VDT is in the *overstrike* mode. This means that a character may be written from the keyboard at the cursor position, and if a character is already located at that position, the new character overstrikes the old. If there is no character at that position, then you are adding on to the existing block of text.

If the "efficient mode" method of storage is used, and there are blank positions at the end of lines, these are disregarded. The text is packed solid from the beginning of the screen to the last character displayed (or possibly to the last character in the buffer). This is even true if "word wrap" is implemented.



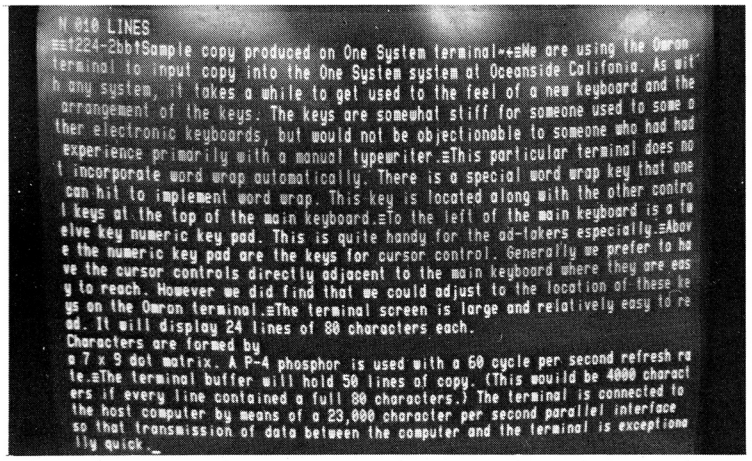
The Omnitext 1500 provides an example of a terminal which does not format display memory in the "efficient" mode. Short lines of text, or lines which have been h&j'd to a short measure (as in this picture) take up just as much display memory (the full 80 characters across) as longer lines.

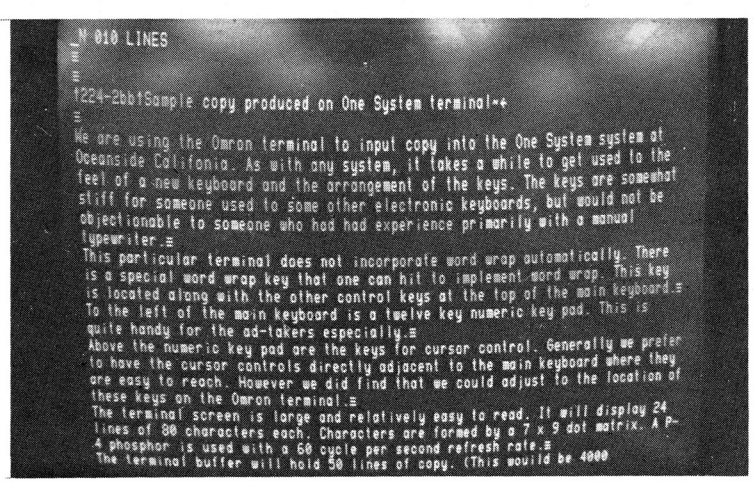
Word wrap.—Most graphic arts VDT's are so designed that they will not break words at the end of a line. If you can display 80 characters across the screen, and if the last word will not fit within these 80 positions, the entire word is moved to the next line (by hardware or software). Except for the space at the end of the preceding line, all remaining locations on that line are effectively disregarded, and you cannot "overstrike" or write within this blanked-out area. Word wrap is not only a convenience. It is a necessity if one is to edit text effectively on a VDT.

In order to get to the character you wish to overstrike, then, you move the cursor to the left or right. The cursor keys "repeat" so that you have merely to hold the key down to cause the cursor to move in the direction you desire. In some cases there is a delay associated with this "repeat" function. If the delay lasts longer than a fraction of a second, this can be annoying, since when the cursor starts to move it may go beyond the desired position.

Some VDT's are so designed that the cursor will move automatically from one line to the next, so long as you continue to hold it down. Others will not move off a given line, and you must strike a cursor "up arrow" or "down arrow" in order to get the cursor to depart from one line and to go on to the next. Generally cursors move only vertically and horizontally, but there are a few VDT's where you can cause the cursor to move diagonally, either by the use of a "track ball" which can be rotated, or by depressing a "cursor right" and a "cursor down" key simultaneously.

Implementation of "word wrap" on a video terminal can take various forms. Some VDT's rewrap every line automatically & continuously as each change affecting line endings occurs. Others use an "open" and "close" approach in which a line or more of space is opened up on the terminal screen, leaving room for changes. Word wrap is then performed, if necessary, on the ensuing lines when the space is closed. Still others (such as the One Systems terminal shown here) will not implement word wrap unless you specifically call for it. *At top:* Unwrapped copy on terminal screen. *Below:* The same text *after* word wrap is implemented.





Delete.—The next common function in "cursor editing" is to delete a character. A delete key is provided, along with other editing keys. By striking "delete" at the cursor position you can eliminate the character in question. Let us consider what happens when you do this.

In the overstrike mode you are, of course, merely substituting one character for another. But when you delete you could either write a "null" or "disregard" code in that position, or you could *erase the character and cause all other characters to move over by one space.* The better systems do the latter, even to the extent of moving up every displayed character on the screen, and re-implementing *word wrap on every line.* This activity takes place so quickly that it is not annoying.

On the other hand, there are some terminals where the process of shifting characters and implementing word wrap is too slow, and shortcuts or compromises are taken. One such compromise would be simply to shorten the line in question when you delete a character, and not attempt to move characters from one line to another. Some systems will move characters upward when you delete, but will not re-implement "word wrap" unless you ask for it by pressing another key which will restructure the data, perhaps after a series of corrections has been made. (One such system calls this key "tidy.")

On many systems it is possible to delete not only a character, but a word, or a line or a sentence. Or perhaps even an entire paragraph. Of course, the delete key has a repeat function (like the cursor keys) and if you hold it down it will continue to swallow up characters as they move into the cursor position from the right. But it may be easier to strike another key captioned "delete word" or yet another called "delete line"—and these, too, can have repeat functions to permit you to make rapid deletions.

For terminals with a "sentence" function, the logic of the terminal (or the external editing logic which supports the terminal) will search for "periods" within the text, and will delete all text from one period to the next. Likewise, a delete paragraph command would delete material from one quad left code (ending a preceding paragraph) to the next.

In some cases, to make sure that you do not destroy text too easily, the delete word, line or paragraph functions may be divided into two parts. One is to define what it is that you intend to do, and then the second portion of the activity is to execute the command. In such systems you may "define word," "define line" or even define sentence or paragraph, causing that amount of text to blink, or to be defined by an underline. Then you can depress the delete key, and whatever it is that you have defined will disappear from the screen and from the terminal buffer.

It may be seen that the delete function is more demanding than the overstrike function since all text below the point of deletion is affected, and may be required to "move up." In the best terminals this restructuring of text occurs instantaneously. It should be obvious that, with such rapid movement of characters, "ghosting" or very slow fading away of previous character images would be annoying. Hence the tubes used generally offer a phosphor with very low "persistence." If the tube is refreshed in 1/60 of a second, then whatever changes are indicated should be effected—at least for a single character—in that time span. Hence successive characters (or sometimes even entire words) can often be deleted in one-sixtieth of a second.

Inserting.—Another important cursor editing function has to do with the insertion of new material. This requires going from an "overstrike" into a "insert" mode. Usually there is an "insert" key. You go "into" insert, add the material you wish—whether one or more keystrokes—and then return to the "normal" overstrike mode again. This may be done in one of several ways—either by the use

of an "end insert" key, or, in some systems, merely by depressing one of the cursor keys.

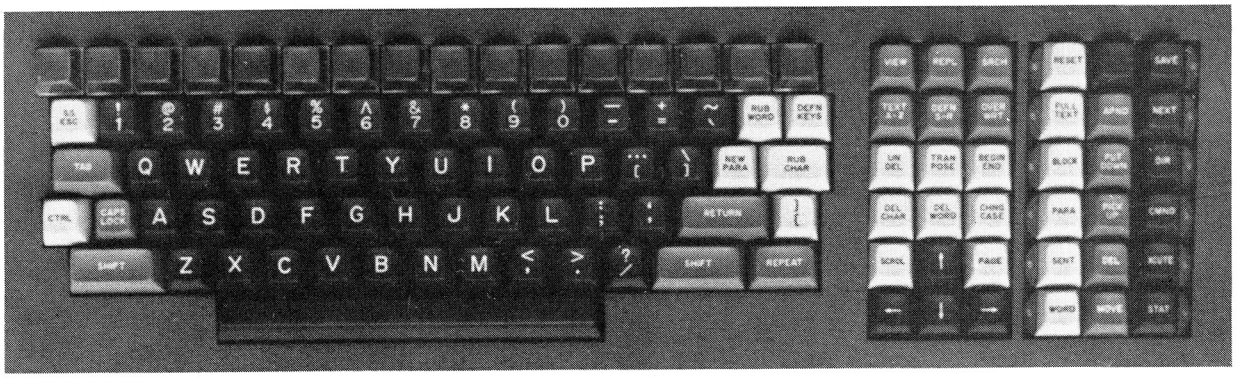
When in the insert mode, material which is added usually precedes the position where the cursor is located. Hence, if you wished the word which appears as "acommodate" to read "accommodate" you would position the cursor either on the "c" and add a "c" immediately in front of it, or on the "o" and add the second "c" in front of the letter "o." You may add not just one character, but several words, or even more text than this. But, just as when you delete text the following material moves left and up to take its place, here the text behind the insertions must move right and downward. Again there are the problems of implementing "word wrap" and of making room for the text to be added. With some terminals, when you strike "insert" the text to the right of the cursor drops down one line, leaving space to add one or more characters. When you strike "end insert" it moves back up again, at the same time implementing word wrap. If the material to be added overflows the line, then another line opens up, with the old material dropping another line to make room.

With other terminals, the material to the right of the insert moves to the right and down with every keystroke added, implementing word wrap on the remainder of the screen whenever and wherever this is required.

In any case, the buffer of the terminal generally provides a limit to the amount of text which can be added. If the buffer is full already, nothing may be added unless the buffer is partially emptied. This may be accomplished by placing the cursor at some point ahead of the location of the desired insertion and striking a "write to cursor" key, which empties out the top of the buffer onto the output paper tape, floppy disc, or sends it on to a computer system, making room to add more text. Usually there is a warning light or "beep" sound when the buffer is almost full, and the keyboard may lock when there is no more room in the buffer. To avoid the overflow problem, when text is "read in" from paper tape or disc, in response to a "read" command, about two-thirds of the buffer is loaded in order to allow room to accommodate any desired insertions.

It may be seen, however, that in the systems such as we have described so far, only limited insertions are contemplated, although by repeated partial emptying of the buffer, through writing out material ahead of the insertion or already keyed as part of the insertion, it is possible to add—somewhat awkwardly perhaps—as much text as may be needed.

A problem may arise, however, if one wishes to make a correction in the material which he is inserting. If he makes a mistake, with some terminals he will have to go out of the insert mode to "delete" or "overstrike" and then go back into the insert mode again. Still other terminals may offer a feature called "rubout" which will permit you to back up, while in insert, and to delete characters from right to left, ahead of the cursor, rather than as the conventional delete works, from left to right, behind the cursor.



Some sample keyboards

Note the similarities and differences in these video terminal keyboard layouts & functions. All 4 have cursor control keys; all have keys whose purpose it is to define and to operate on an element of text.

Top: DEC's VT71/t.

Next: Atex's reporter input terminal.

Right: The Hendrix 5700 keyboard.

Below: Keyboard of the Varisystems Vari-Composer.







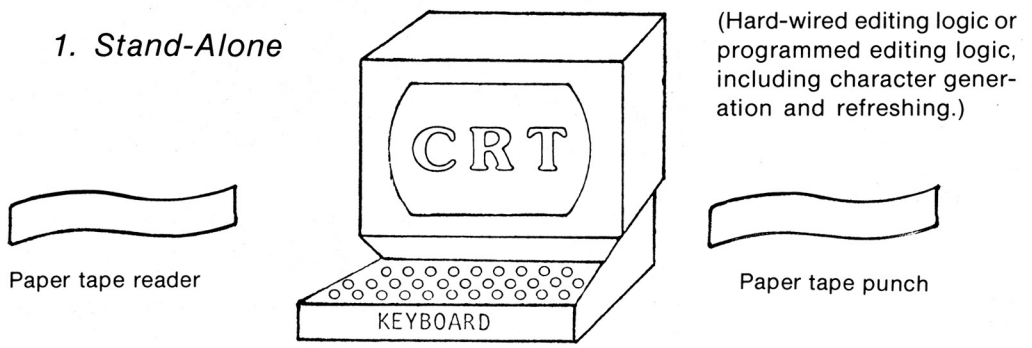
Other editing features

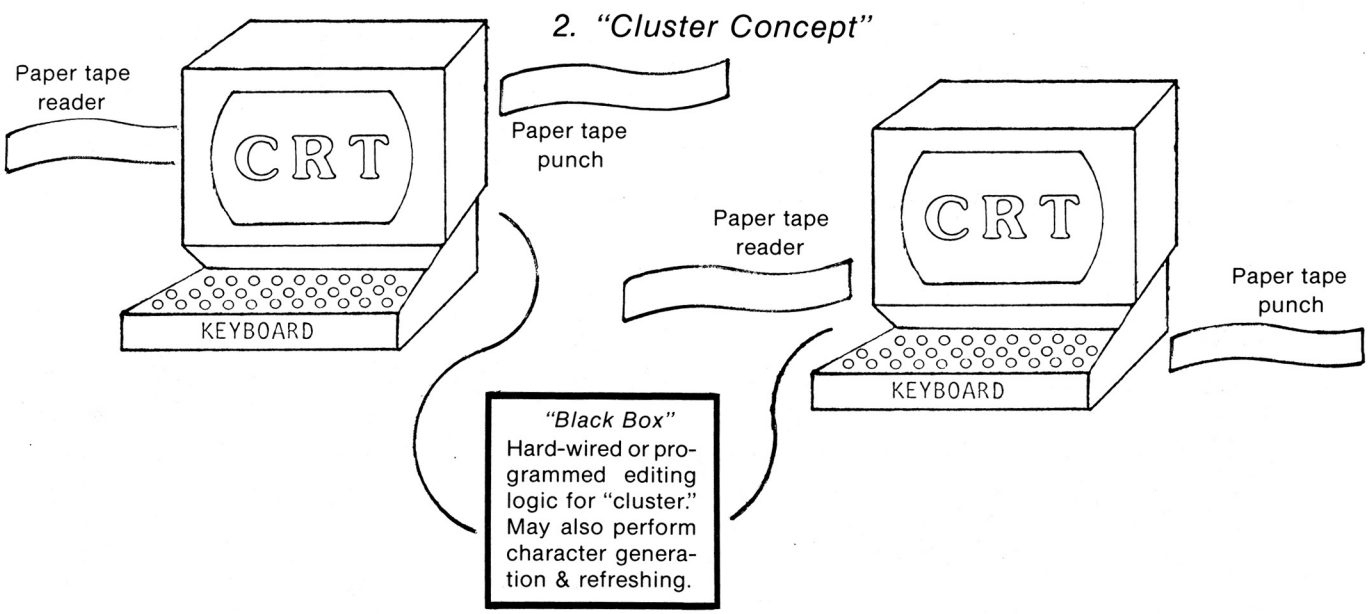
We have described the most common editing features: overstrike, delete, and insert. There may be others: "copy," "move," "store," and even "search" or "search and replace." But before we discuss these, we must give a little more thought to the functional characteristics of the terminal itself.

Terminal architecture

We began our discussion of terminals with a description of the so-called "stand-alone" VDT. Input came from paper tape which was read in—perhaps several paragraphs or a buffer-full at a time. Output went to the paper tape punch in the same manner, preparing a new, "clean" tape which could then be mounted on the paper tape reader of the phototypesetter, or fed into a computer system. More recent models substitute one or more floppy discs for the paper tape reader and punch.

Video terminals may be configured in a variety of different ways with respect to their computer controllers. Here are two specific examples of possible terminal/controller configurations.





Paper tape reader/punches could be attached to "black box."

With such a terminal, the character-generating electronics, the refresher circuitry, and the editing logic itself must all be built into the single device. It is, as we said, a "stand-alone" unit.

But it is also possible to configure the terminal in some other way. In fact, there may be a number of different approaches to terminal design.

For example, you could have a "cluster" of two or more terminals. Each terminal may have its own paper tape reader and its own paper tape punch or its own floppy disc controller, but the character generating and refreshing logic, as well as all of the editing logic, could be provided by a supporting "black box." Or each terminal might have its character generating and refreshing capability, and the editing logic could be borrowed from the black box. It may well be more economical to use the same electronics and even the same character generating and refresh capability to service more than one terminal where several terminals are intended to be used at the same location.

The "logic" which performs the editing could be "hard-wired"—that is, not programmable but built to perform the defined tasks. Or a mini- or microcomputer could be programmed to perform these functions and even—perhaps—the character refreshing function. It has become more and more common to use programmable computers for these purposes, and consequently suppliers have found it possible to utilize the same computer for the performance of other functions as well.

The counting keyboard/editing terminal

One such application contemplates that the same terminal could be used not just to correct material which has already been keyboarded, but for initial keyboarding as well, and not merely for the keyboarding of unjustified text, but to furnish counting logic so that the keyboarder can make appropriate end-of-line decisions. In other words, the terminal will store width values for the desired type faces, add up these values and signal when the operator has brought the line within justification range, and again to indicate when the line has been overset, so that the last word or portion thereof can be deleted.

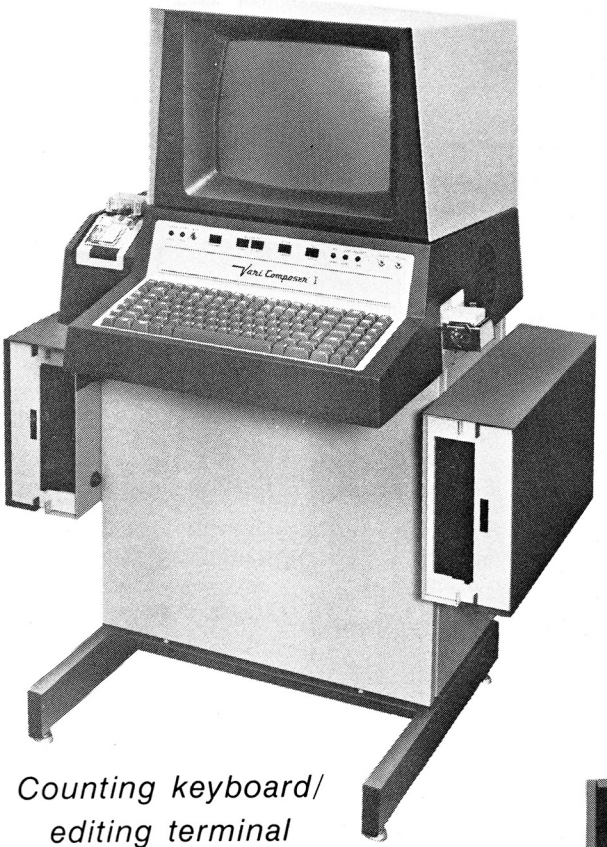
On other occasions, the terminal could be used to read in paper tape (or text from floppy discs) keyboarded elsewhere, simply for the editing of the unjustified text, or perhaps for the insertion of typographic commands.

Or unjustified text could first be input or read in, and then corrected. Subsequently, the counting logic could be applied to the incoming text, on a line-by-line basis, so that the corrected output would also now contain end-of-line decisions.

The next step in the evolution of the video terminal was to allow the operator to key or read in text in an automatic mode, whereby those lines that would "fit" would be output with their end-of-line signals inserted by the logic of the VDT.

For those lines that could not be justified, the cursor would position itself on the last character (within the overflowing word) which could be fitted into the line. The operator may then decide if and where to hyphenate the word in question, making a manual end-of-line decision, and returning control to the automatic mode until another overset line is encountered.

VariComposer



And then, finally, simple hyphenation logic was built into some VDT's so that the process could be completely automatic. Unjustified input can be read in from paper tape or diskette, the text can be corrected and updated, and then it can be justified and hyphenated and prepared with the desired codes for a particular typesetter.

Moreover, to facilitate the process of making corrections, a "search" feature is built into the VDT, so that the operator can work his way from one correction to the next, allowing the VDT to process intervening text without interruption, but requiring that it stop when it encounters a certain combination of consecutive characters or codes, to permit operator activity.

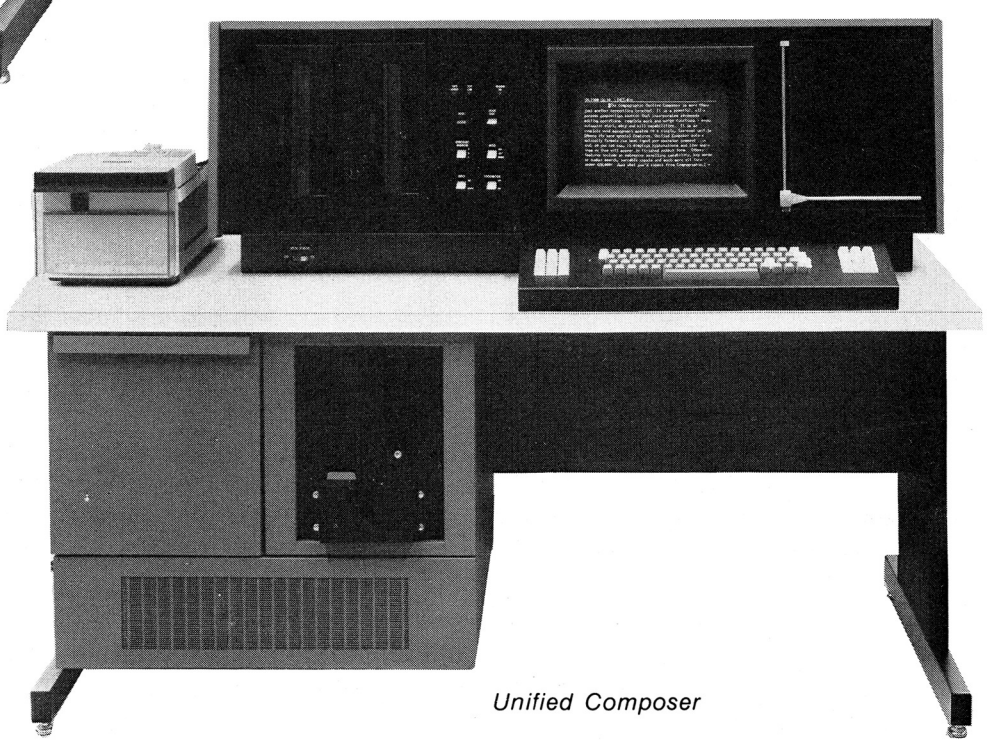
We have oversimplified the presentation of this evolution. It is not easy to use the same VDT simultaneously as a counting keyboard and as an editing terminal. It is not so difficult to use the VDT as *one or the*

Counting keyboard/ editing terminal

Both the Unified Composer at right, developed and marketed by Compugraphic Corp., and the VariComposer, a product of Bobst Graphic/Varisystems, are stand-alone *counting* terminals. They are both floppy disc-oriented devices, accept input in a variety of forms, and produce justified output to drive a phototypesetter.

Either terminal may be used in manual, semi-automatic or automatic modes of hyphenation. In the latter case, the terminal uses a simple hyphenation algorithm to determine hyphenation points.

Such devices are offered by other vendors as well.



Unified Composer

other. The problem arises if you wish to make a correction in a line which has already gone through the counting logic calculation. Obviously, if you go back and insert or delete material within that line, it will have to be recounted. You may be able to proceed to make corrections more or less on a line-for-line basis, but if the screen contains a good many lines, most of which have already been processed but not yet output, it is usually not possible to "back up" and make a correction at the top of the screen and to reprocess the subsequent material.

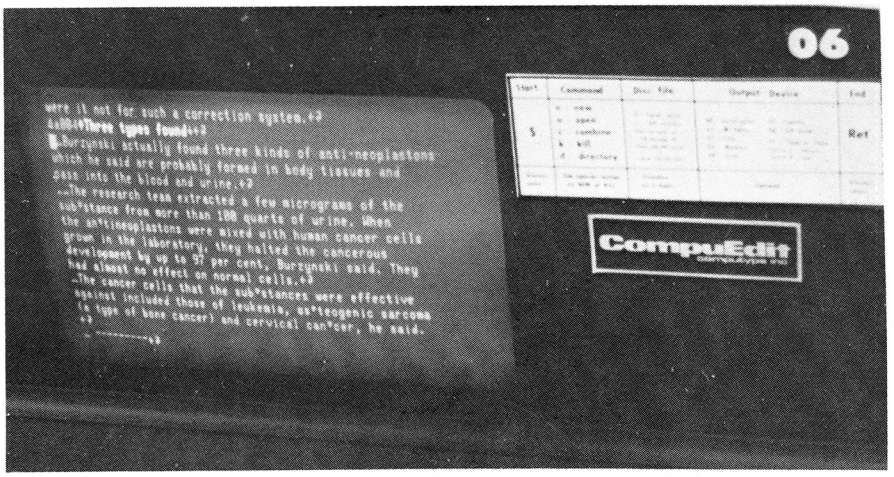
Nevertheless, with the use of video terminals the industry has become more accustomed to the convenience of the "soft copy" display. Input keyboards began to look—and even to act—more and more like editing terminals. And even where video terminals are not used, we now find that many input keyboards offer certain editing features. A small self-scan display may be provided, along with a search feature, so that previously-prepared input may be read in, perfected, and written out. But generally speaking, it is only the VDT which offers the kind of "buffer" within which corrections need not be made sequentially—and only then, as we have noted—if the counting or justifying process is not taking place simultaneously. It may be possible, however, to use the terminal first to key original material, second to correct it, and third to justify it, all on the same device but as three different processing steps—perhaps calling for three different modes of operation from the terminal itself.

The "move" function

Returning, now, to a discussion of editing features (and disregarding the possible use of the terminal for counting or assisting with end-of-line decisions), we can consider the "move" command as illustrative of still another editing capability.

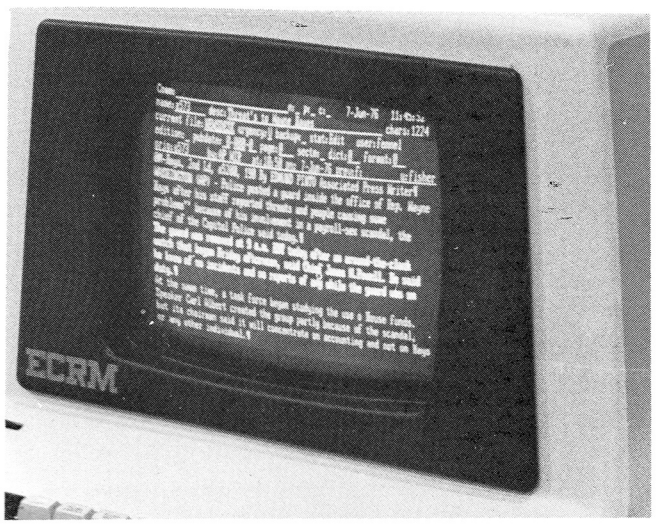
It is often convenient to move text about on the screen. This is more likely to be desired when the terminal is used for "editorial" purposes—when a writer or editor is manipulating his own text, and perhaps creating it as he goes along, using the terminal for initial input. Even without a "move" command, it is possible to move text—or to achieve the same results—by deleting material from one place and inserting it at another. The difficulty with this is that one may not remember what it is one has deleted. And obviously this procedure requires more effort. It is clearly more convenient to be able to "define" a particular word, a phrase, a sentence, a paragraph, or some other block of copy (and there is often a "define block" command which permits the operator to signal the beginning and the end of the text he wishes to manipulate). Then the cursor can be positioned at some other location, and the block so defined can be "moved" to the new cursor location, whether it is to a position earlier in the text, or further down the line.

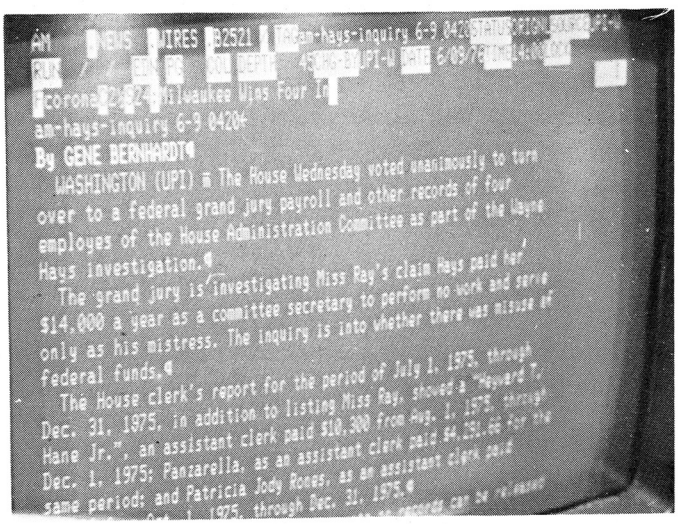
If only a few words are to be moved, and if they are only to be moved a short distance, it may be possible to make such a move within the area of text displayed on the screen. But often one desires to move text to a location which lies outside the displayable area. This may be accomplished by *forward or backward scrolling* and is possible if the buffer is larger than the amount of text displayed at one



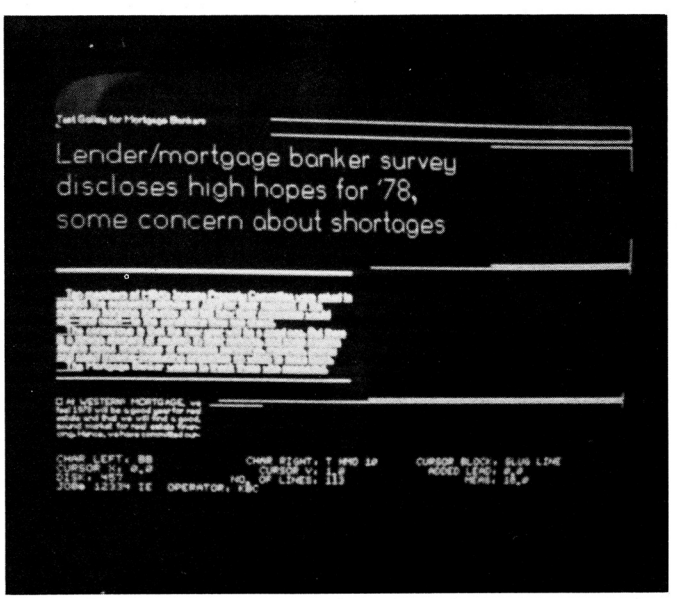
Left: Even quite simple terminals, such as Computype's CompuEdit, are often capable of displaying text in more than one mode. Here an emboldened mode is used to differentiate a subhead from the body text.

Below: The Amcomp system (since acquired by Logicon-Intercomp) added a reverse mode to the normal and bold. An example of it can be seen in the header material at the top of the display.

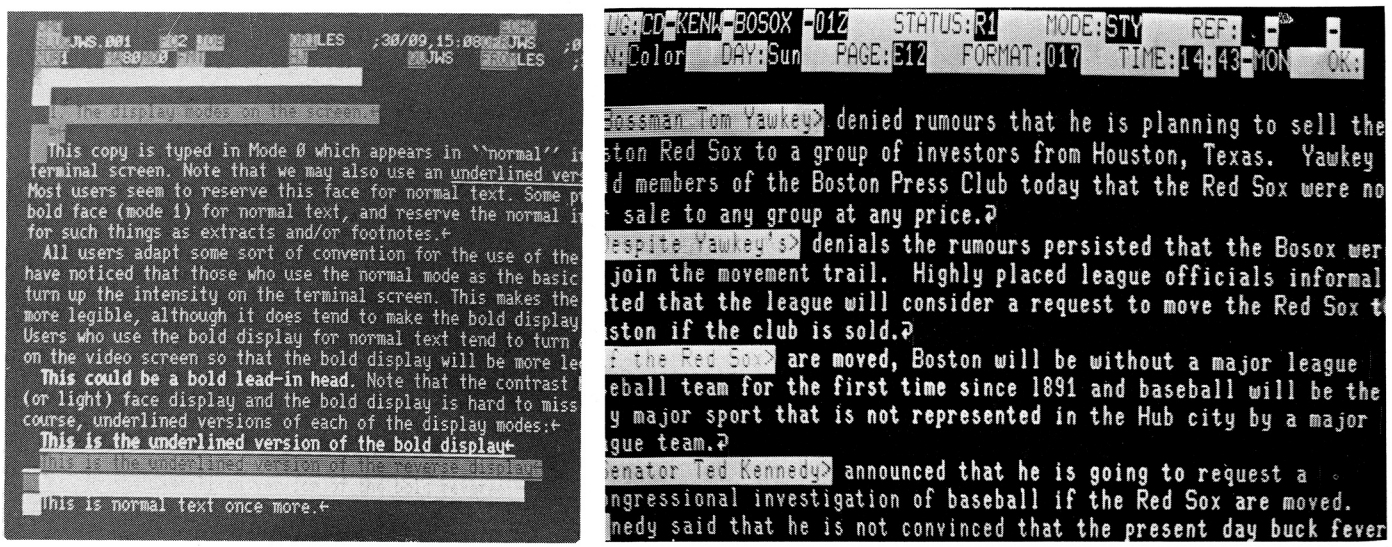




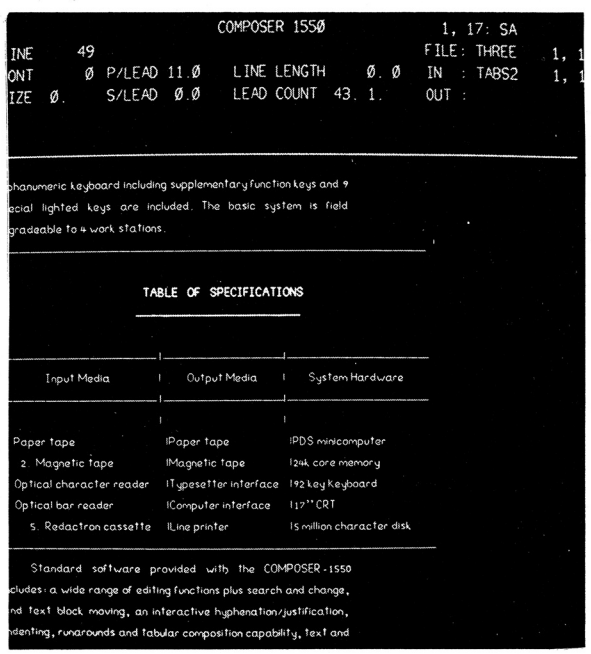
Above: ECRM's 7600 system terminal adds an underline feature to aid in further differentiating display modes. These different types of character displays can be used to correspond to different type faces or fonts, or they can highlight terminal functions. Often, as on the display pictured here, any block of text which has been defined for some purpose (such as moving or deleting) will be temporarily emboldened. Sometimes a defined element will be made to blink.

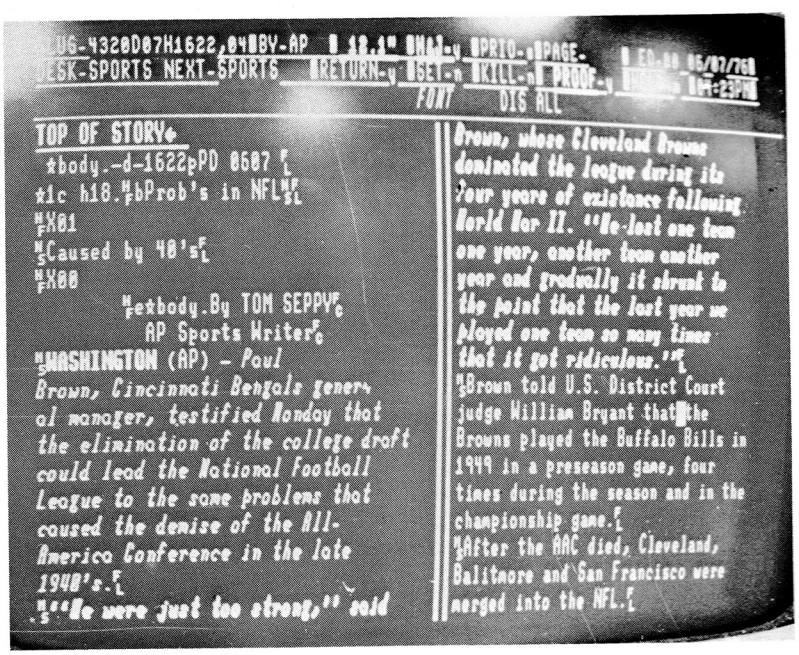


Right: There are some very special VDT's which have greater power and flexibility. This one shows simulated type in the actual point size and with true character widths, and it permits the user to perform "area composition." It is the Syntext system, developed by N. Edward Berg.



Left: The Atex terminal features eight different display modes; each can be used to represent a different output type face and/or type size. *Right:* The VDT used on the Hendrix 3400 system makes distinctive use of reverse character display.





Left: Imlac uses a random-stroking technology, rather than a raster scan technique, in order to represent proportionally spaced characters on the display. The use of proportional characters, in conjunction with the 1550 system's horizontal scrolling capability, makes the terminal an ideal tool for the formatting of tabular material.

Right: Composition Systems, Inc. (CSI) uses a "soft display" technique for its CSI 103 terminal. Different programs and character sets can be loaded from the host computer. In this instance, it is shown using a double-column display format with both italicized and roman characters appearing on the screen.

time. For example, if the terminal buffer contains 10,000 characters, of which only 2,000 can be displayed at once, it may be possible to define the desired sentence or a group of words, move the cursor forward to another location within the 10,000-character take, scrolling in that direction, and place the defined material at the new position. Or, perhaps, the text to be moved appears toward the bottom of the buffer. One can reverse scroll and insert it somewhere earlier within the text stream. Consequently, there is a connection between the *act of scrolling* and the *act of moving material from one location to another.*

There are other ways in which "moves" can be made, but they are often quite awkward. For example, one could write out the paragraph or block of text to be moved, and then (if there are two paper tape readers on the VDT, or different files in the floppy diskette) merge it at the desired location. Obviously this is not so convenient, and yet it may be the only way to handle block moves if they exceed the buffer of the terminal itself.

The "search" function

The same thing may be true of the "search" function. It may be possible to search through a buffer-full of text for the occurrence of certain words or character combinations. It may be possible to issue a command to substitute one string for another—but this cannot be done except as a separate processing step (reading in and punching out) unless it is done only within the terminal buffer. And so we find that there are many terminals which limit the "move" and "search" capabilities to the amount of material which can be contained within the buffer of the terminal itself—if the move and search functions are to be executed along with other functions, such as randomly inserting and deleting material.

We can see, therefore, that editing power is much greater if it is possible for the VDT operator to scroll through a large quantity of text—rolling the text backward or forward on his screen. But since the terminal buffer is limited in size (if there is one at all), it would be most convenient to be able to call the material from storage, roll it through the window of the VDT, and store it away again, doing this as often as one might wish, provided that the movement of text into and out of the terminal does not require too much waiting time on the part of the operator.

The on-line VDT

This brings us, then, to a consideration of how it might be possible to "interface" the VDT to a computer system, with the text storage which that system offers, and to avoid the arm's-length relationship which a stand-alone VDT—or a stand-alone cluster system of VDT's—requires. If this can be accomplished, perhaps the scrolling can be "infinite" and not merely related to the size of the buffer.

"Bit serial" information transfer.—One common way to "interface" or connect a VDT to a computer system is to arrange for the text material to be transferred to and from the VDT by means of a "bit serial" mode of transmission. What this

means is that a bit stream is communicated from computer to terminal and back to computer again, and that these bits are arranged into "bytes" or "words" when they reach the terminal or when they are returned to the computer. Often such interfaces make use of a phone line, and the information is related as auditory signals. The speed of such transfer is described as the "baud" rate. At 2400 baud, 2400 bits of information may be transferred in one second's time, and this is roughly equivalent to 240 characters, since up to 10 bits may be required to describe each character in a bit-serial transmission mode. To fill a 2,000-character buffer and video screen would thus require perhaps eight seconds of transmission time. To dispose of that information (sending it back, corrected, to the computer) and to get another 2,000 characters would thus require more than 16 seconds. If the buffer contained 4,000 characters, each transaction would take twice as long.

Paging.—Usually, when you interface a terminal by bit-serial connection, you "page" text from disc to computer to VDT, and back again. A "page" is a terminal "buffer-full" or some portion thereof. If the editing logic is built into the terminal, then only that material within the "page" is subject to correction. And if you wish to access text buried deep in a file, you may have to page through many buffer-fulls—a very time-consuming process—to get at the text you wish to work with.

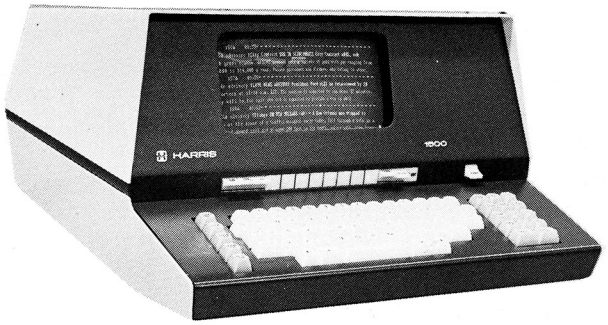
One way to speed up this process is by using a higher baud rate, if it is available—such as 9,600 baud, or even 19,200. At the latter rate a 2,000-character screen can be filled in about one second.

To avoid "paging" it is possible to "program the computer" so that some of the editing functions may be performed by computer logic rather than by terminal logic—especially the ability to "get at" or "get to" a particular line (identified by line number), or by contextual search—and to bring in only those "pages" which need to be edited.

The computer main frame may also be used to implement "moves" that cannot be effected within the buffer. The material to be moved can be sent back to computer memory or disc and recalled when the desired "page" is on the terminal, to be inserted at the indicated location.

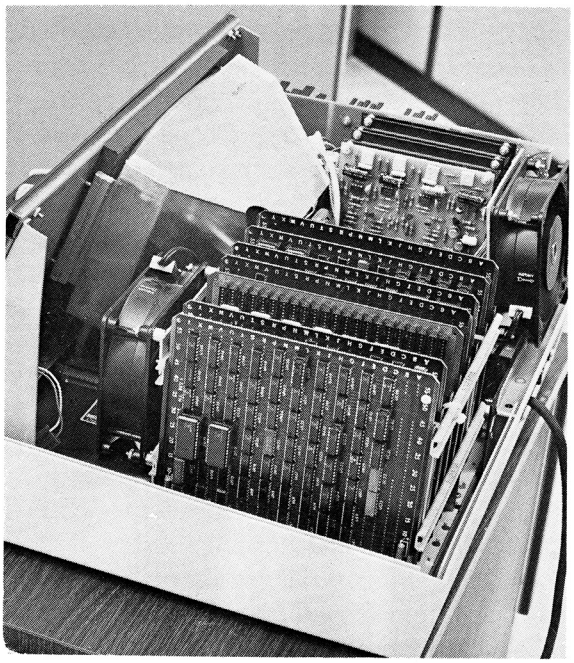
Finally, even with a bit serial interface, it may be possible to scroll quite rapidly through an entire file, by "paging" a line at a time. Even at 2400 baud, a line of 80 characters may be filed away, and another line fetched, in less than a second. Then text (within certain limits prescribed by the size of the buffer) could be "tucked away" within the terminal and re-inserted elsewhere, as a result of a forward or backward scroll.

Parallel interface.—Another solution is to transmit characters in a "parallel" fashion rather than "bit serially." This means that eight- or nine-bit bytes can be transferred at comparable bit transfer speeds. A 9600 baud parallel interface would make it possible to transfer perhaps as many as 9600 characters per second, or to dispose of one 2,000-character screenful and to fetch another in less than a half second.

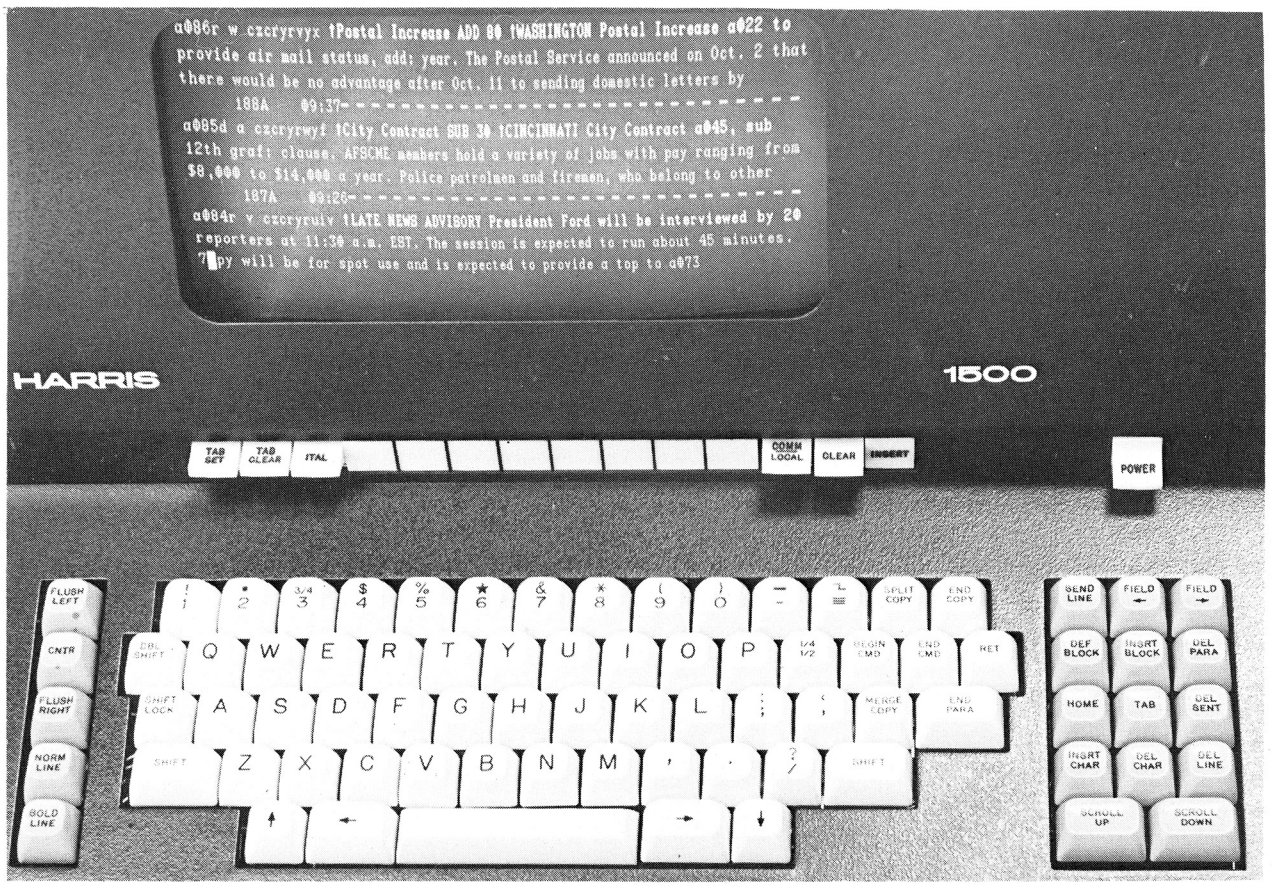


The anatomy of a terminal

The Harris 1500 terminal (used in Harris 2500 systems) is a self-contained unit with its own buffer memory, its own refresh and character-generating logic, and its own editing logic. The display on the screen consists of 12 lines of up to 80 characters each. Characters are formed by a 7 x 9 dot matrix. A total of 112 different symbols may be displayed. The buffer memory contains only one "screen" (12 lines) worth of information. The host computer supports the terminal by providing scrolling and block move functions. The 1500 keyboard (below) is modeled after that of a standard office typewriter. Note the keys to control cursor movement at the bottom of the keyboard on either side of the spacebar. Simple editing functions are controlled by keys on the keyboard to the right of the main keybank.



Above: The Harris 1500 terminal with its cover off. Notice the amount of electronics, and the accessibility of the components.



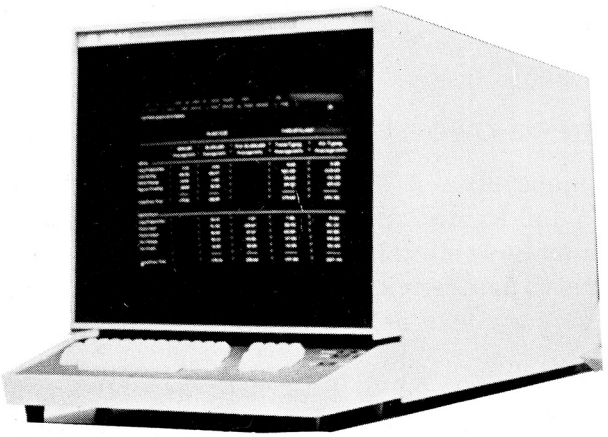




Video display terminals come in all shapes and sizes

Above: The *Teleram P-1800* terminal is a portable VDT with magnetic cassette storage. It is small enough to fit under the seat of an airplane.

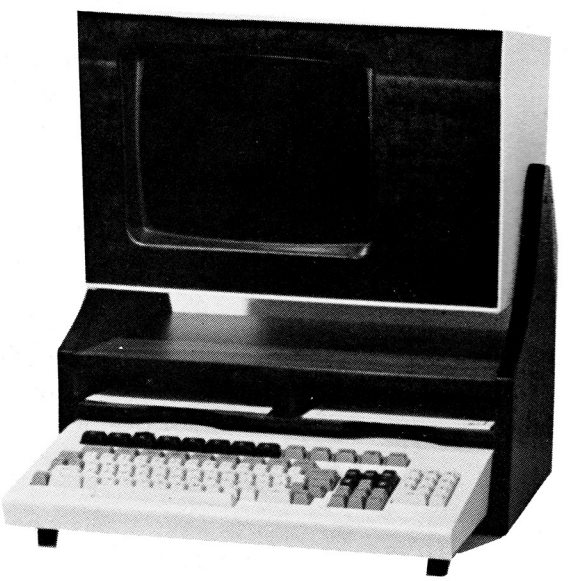
Right: Imlac's Composer 1550 offers proportionally spaced *characters generated on a rel*atively large display area of random-stroking tube.



Above: Tal-Star's dual-screen terminal facilitates editing and merging of two separate files.

Below: The display on *ONE Systems'* terminal (on left) is *raised to the operator's eye* level, while the smaller *Mycrotek* tube (on right), similarly elevated, offers cubby holes for desk storage and a movable keyboard.





The DMA solution.—A third approach is to interface the terminal in such fashion that the buffer does not reside in the VDT at all, but in a portion of computer memory. Under such circumstances, the character generation and refreshing may also be done by the computer, and the VDT has no "intelligence" of its own. *All* editing logic is performed by the computer, and the VDT screen is merely a "window" into a section of computer memory. The keyboard, separately connected, can directly address computer memory on an "interrupt" basis, and response times are often at computer speeds, reflecting themselves in the implementation of the desired change in one "refresh cycle." Infinite scrolling (or "virtual" scrolling, as it is sometimes called) is usually a feature of such systems. The penalty you pay is that, generally speaking, such VDT's may not be located remotely, but must be within a thousand feet or so from the main frame computer. The other "penalty" is that the more intimate—rather than arm's length—relationship between VDT and computer requires more support from the computer itself (thus imposing more of a load, and, under some circumstances, bringing about a deterioration in computer response time or overall performance). More computer "overhead" is also usually involved.

The trend away from stand-alone VDT's

Unquestionably the trend has moved decidedly away from stand-alone VDT's and toward an intimate relationship between terminal and computer, or computer system. This makes it possible for many terminals to access the same files and to pass "stories" from one writer to another, so that the computer system handles *copy flow* as well as copy processing.

On the other hand, to facilitate response and to reduce computer overhead, there is a tendency to "distribute" the processing intelligence, so that intelligent terminals with their own microprocessors may be used, and they, in turn, may relate to specialized controllers, one of which may handle I/O (system input and output and file management), another may perform h&j, still another may cope with those editing functions which lie beyond the capacity of the microprocessed VDT's, and, finally, a different computer may be employed to perform page make-up functions.

VDT's as part of a system concept

Thus we have moved, especially since about 1974, more and more in the direction of "systems" which utilize on-line VDT's for input, for editing, for interacting to and modifying computer decisions with respect to h&j or page make-up, for communicating with others involved in the production or editorial process, and for diagnosis of problem jobs which otherwise might flow through the system unattended.

We shall be considering some of these arrangements and systems in the next chapter.

Review Questions:

1. What is a terminal "buffer"? Discuss some of the implications of having a small display buffer, a large buffer, or no buffer at all with respect to terminal responsiveness and editing functions.
2. What is meant by the term "cursor editing"? Cite some examples of editing functions which are illustrative of this mode of operation.
3. What is the difference between a "hard-wired" and a programmable terminal?
4. Enumerate some of the factors which would affect the quality of a video terminal display.
5. What is meant by the phrase "efficient mode of storage" when applied to a VDT display?
6. Describe some of the different methods used by terminal manufacturers in situating the character generation, refresh and editing logic portions of memory with respect to the display itself.
7. Enumerate the most common types of editing function keys generally found on a graphic arts video terminal keyboard and describe a series of operations involving their use.
8. Aside from the conventional TV raster scan technique, what other types of display technologies do you know of that are used (or could be used) in VDT's for the graphic arts?
9. How might different display modes on a terminal aid in editing, formatting, or composition functions?
10. What is "paging"? What is "scrolling"? How do they differ?
11. In what types of applications would "virtual" or "infinite" scrolling be a necessary or desirable attribute of a VDT system? When would it not be so necessary?
12. What are the implications of having a bit serial as opposed to a bit parallel interface between a terminal and its computer?

19

Disc Systems and File Management Schemes

WHEN VIDEO DISPLAY TERMINALS are interfaced to a computer so that they can fetch text from computer storage—or create text and file it there—and so that operators of the terminals can engage in editing and other text processing activities, it is necessary to develop techniques to "manage" the textual material in question. These are tasks similar in many respects to those that computer systems analysts must cope with for any "on-line" and "time-shared" or "real-time" system involving multi-processing or multi-tasking activites.

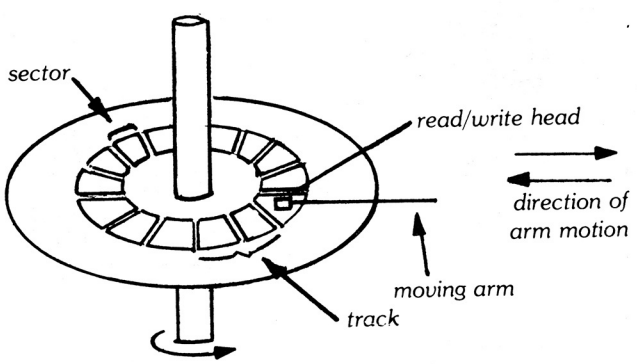
To understand how data files are managed it is necessary for us to know more about computer storage techniques.

Computer memory itself is not sufficient to offer room for the storage of large quantities of text. The core or "memory" is used for the temporary manipulation of this text—providing the programs, look-up fables, indexes and other necessary processing techniques along with buffer areas within which portions of the text temporarily reside. The text itself—as long as it stays within the system—is generally stored on one or more moving head discs, although magnetic tapes also occasionally serve the same or a similar purpose.

There are *fixed-head* discs—which are more expensive, and generally offer less storage, but faster access times. *Moving-head* discs are more commonly used. In either case the information is stored magnetically on tracks. It is written by a "write" head which moves over to an available track and records data as bytes or characters (really bit patterns). It is read by a "read" head which functions in the same manner. (In fact, the same head serves read/write functions.) A moving-head disc may consist of a single platter or of several platters. It is mounted on a disc drive, which is governed or directed by a disc *controller*. The controller, under a general operating system and a file management system, tells the write head where to record data and the read head where to find it. A multiple platter disc works in the same fashion, except that there are more read/write heads—one for each useable surface of the disc.

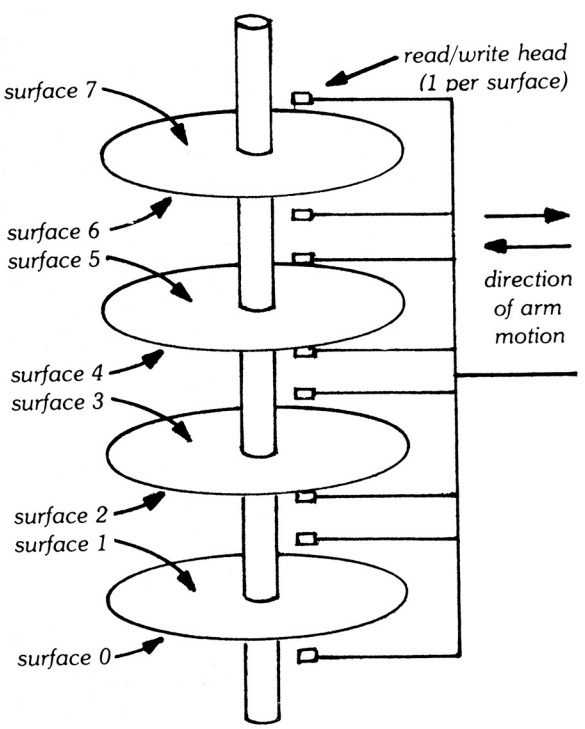


A multiple platter, removable moving-head disc mounted on its disc drive. The plastic cover protects the rigid disc from being damaged by dust particles.



A moving-head disc

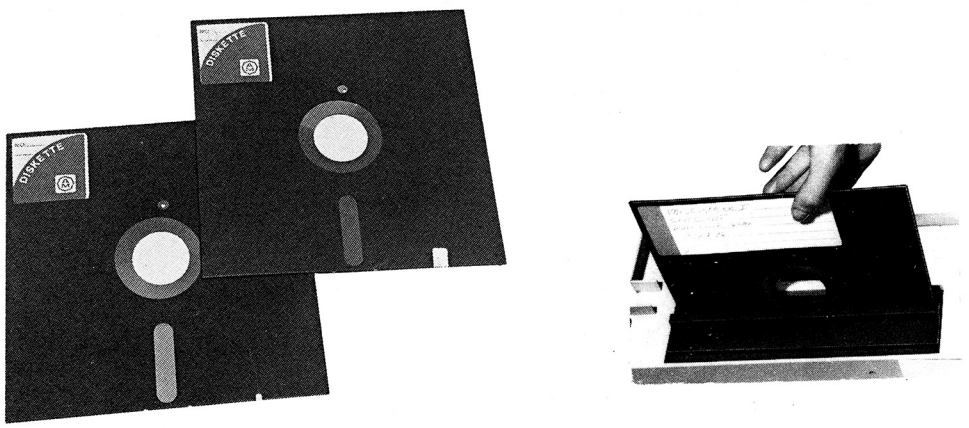
Information is recorded onto the magnetized surface of the disc on a number of concentric circles, called *tracks*. There are usually several hundred tracks per disc surface. The movable "head" is wide enough to read information from one track at a time. The disc tracks are divided into *sectors*. There are usually anywhere between 10 and 100 sectors per track. In a multiple platter disc setup, the moving arm supports one read/write head for each disc surface. They all move in and out simultaneously, under control of the disc drive controller.



A certain amount of data can be written to each track of a disc. For example, a Digital Equipment RK05 disc and disc drive (with RK11 controller) provides a cartridge which holds something over 1.2 million 16-bit words (2.5 million characters). There are two surfaces, totalling 200 tracks (plus three spares), 12 sectors per track, with each sector recording 256 16-bit words. A one-track move takes 10 milliseconds, average track positioning time is 50 milliseconds, and maximum is 85 milliseconds. Data transfer speed is 11.1 microseconds per word; disc rotation speed is at 1500 rpm, and the time for a one-half revolution is 20 milliseconds.

We can see, then, that a track can be accessed, on the average, in .005 seconds, and that an additional 20 milliseconds or so might be required before the desired sector is under the read head. At that time 512 characters can be brought into memory at a data transfer speed of 11.1 microseconds per word, or about 5 microseconds per character (.000005 seconds). A double-density disc might read or record twice the amount of information in the same time.

The read/write head does not come into physical contact with the disc (unless it is a "floppy disc" or diskette). The head floats over the track on a cushion of air created by the spinning motion of the disc. Consequently, these "rigid" discs do not wear out, or are not damaged unless there is a "disc crash" when the head comes into physical contact and destroys data. Otherwise, when the system is shut down and the disc ceases to spin, the head retracts.



Left: Floppy discs. *Right:* A "floppy" being inserted in its disc drive.

Each track has an address, and within each track there is also a sector address. But the information on a given sector—say 512 characters—can only be read when that sector spins by the read head, and the head is positioned over the proper track.

The information stored on a disc is not generally written consecutively, but rather randomly, in order to optimize retrieval time. Data may start out by being more or less consecutive, but if more information is added, then the text will overflow the allotted area and the extra (left over) data will be written elsewhere—any place that is available on the disc or disc pack, leaving behind an address

```
DATE: 09-16-76      JOB: ksb      PACK: SYSTEM

   FILE        FILE  USE    ----------DATE----------
   NAME        SIZE  CNT    CREATE    ACCESS    CHANGE
-----------    ----  ----   --------  --------  --------

THERE ARE NO EXISTING FILES

DATE: 09-16-76      JOB: ksb      PACK: PACK1

   FILE        FILE  USE    ----------DATE----------
   NAME        SIZE  CNT    CREATE    ACCESS    CHANGE
-----------    ----  ----   --------  --------  --------

pr0loptns      0002  0003   08-05-76  08-11-76  08-11-76
pr0lsyscon     0001  0006   08-05-76  00-00-00  08-09-76
pr0locr-sp1    0001  0009   08-17-76  08-19-76  08-19-76
pr0lmod&con    0003  0003   08-11-76  08-16-76  08-11-76
ad0lpublist    0005  0006   08-06-76  08-06-76  08-06-76
pr0ltrain      0002  0008   08-05-76  08-19-76  08-11-76
pr0lrecom1     0003  0003   08-05-76  08-09-76  08-09-76
pr0lrecom2     0001  0002   08-05-76  08-09-76  08-05-76
pr0lresplet    0001  0005   08-06-76  08-06-76  08-06-76
pr0lkeybd-1    0003  0006   08-06-76  00-00-00  08-06-76
pr0lprint-sp   0002  0006   08-06-76  08-16-76  08-16-76
pr011500-sp2   0002  0008   08-06-76  08-11-76  08-06-76
pr011500-sp3   0001  0003   08-06-76  08-11-76  08-06-76
pr011500-sp1   0001  0008   08-06-76  00-00-00  08-06-76
pr0locr        0003  0011   08-11-76  08-19-76  08-11-76
pr0lkeybd-2    0003  0007   08-06-76  08-16-76  08-16-76
pr0lkeybd-3    0003  0008   08-06-76  00-00-00  00-00-00
pr0lmss        0002  0003   08-06-76  08-11-76  08-09-76
```

Many typesetting systems offer fairly sophisticated file management capabilities in both floppy and rigid disc system configurations. In the system shown here, there is no limit to the number of "jobs" or of "files" within jobs which may be contained in disc storage except for the obvious limitation of filling the disc to capacity. As files are created, or input into the system, they are assigned a file name of up to 12 characters in length. The operator also indicates the name of the job (or in-basket) to which that file is assigned. In order to access a specific file the operator must call for it by job name and file name; if he or she cannot remember the job name to which the file was assigned, then an "all job" directory may be invoked. The job directory (pictured here) lists the file names, the sizes of the files (in thousands of characters), the number of times the job has been used, the date the file was created, the last date the file was accessed, and the date of the last change to the file.

indicating where to go next (what track and what sector or platter). The supervisory program and controller may also keep a record of where the *previous* batch of information was written, so that a "file" can be recovered by tracing it in either direction, once you know where to go to get an identifiable piece of that particular file. A "bit map" tells the disc controller where there are unused areas on the disc. An index—either in memory or on a designated disc area—will indicate where the read head should go to pick up the beginning of the file.

Consequently, a name must be associated with a particular file, and an index must be constructed, showing where that file's beginning may be found. If you want to retrieve the file you need to tell the file management system the name of the file that you wish to retrieve.

More simple-minded systems may place the burden of keeping track of files on the system operator, in which case he would have to divide up his job into small takes and direct the system to go to a particular track (which he will have written down on a slip of paper) in order to get at the desired job. A higher degree of sophistication would keep an index on the location of the beginning of his file, but he will have to remember the name of the file.

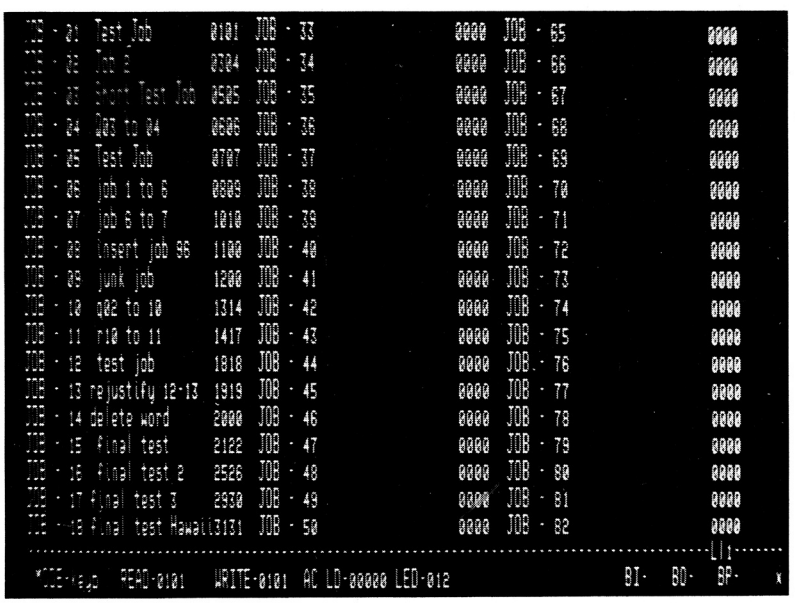
Disc-oriented file management systems can be—and are—very sophisticated. Yet without such systems it is not possible to command the computer to fetch a

file, or to add to it, or to store it, without destroying other files. And without an index, and a trail which describes the current sequence of the file, it would not be possible to pull out the text in order to scroll backward or forward through the file, by bringing it from disc to memory and from memory to terminal.

The use of directories

Sophisticated systems do not expect the user to remember his file names or to know where the files are written, although if the disc platters or packs are removable he will have to know which disc or pack is required. Directories will be provided so that the user can ask to see the names of all files, and their current status. He may even be able to access the desired file simply by placing his cursor on the file name and issuing a "get" command.

But a large system will require the storage of many files, and a file directory—no matter how organized—may be too long to peruse. So, other techniques are developed. One is to search the directory to locate the file, if you happen to remember the file name or certain of its attributes (such as that it was one you created). Another is to create sub-directories, associated with a particular user, or a class of work, such as "sports" or "wire service" or jobs for customer "Jones." Newer systems may offer "cross directories" so that it is possible to search several categories in order to find, say, all sports stories filed by local reporters between noon Monday and 7 a.m. Tuesday—or even just those within this category which have been accepted for publication and have already been typeset. Some systems will enable you, on demand, to construct a listing of all versions of the same file, indicating who worked on that file, and in what sequence changes were made. Some of them can even indicate, by means of an "audit trail," what specific changes were introduced at each phase in the editing process.



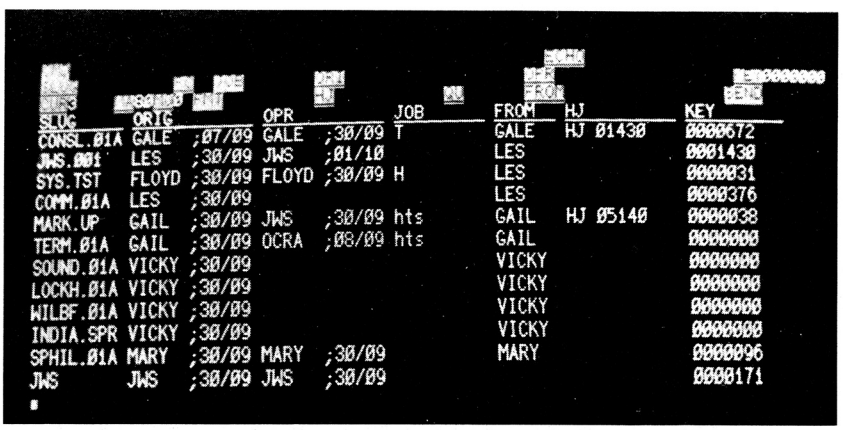
Varisystems' VariComposer 1A, a floppy disc-based, single terminal, editing/composition "system," has a very basic file management system, which requires the operator to perform some of the file management functions which are performed automatically on most larger systems. Here you see a listing of all of the jobs stored on the floppy disc.
Each job area is numbered from 01 to 82. In the space to the right of each job number, the operator fills in the name or number of the job he or she is beginning. The operator then fills in the beginning track number for each job. If the job exceeds the capacity of a single track (4,096 characters), it will be written onto the next sequential disc track, whether or not other material has already been stored on that track. It is up to the operator to manage the allocation of disc tracks for each job. Files can be "protected" to prevent overwriting, and the operator is warned that he must make some other disposition of the overflowing text.

Routing files

If there are many terminals on the system, it may be possible to "send" a story from one directory to another. Thus a reporter may "file" his story and direct it to his editor who, after reviewing it, will send it to the "slot man" and he, in turn, will assign it to a particular page. In each instance the story will be shifted, or copied, from one directory to another, although *there is no physical transfer of the information* unless the file is copied. It is the "pointers" to the file which are changed. Some systems will provide "default" routines which will automatically route stories in a certain sequence and stipulate whether the directory to which they are sent will list them according to length, or put the newest or oldest items on the top of the pile. Thus the directory serves as an "in basket" which is always maintained in a certain order. Messages can even be sent to the receiver, and a "message waiting" indicator may flash on his screen so that he will know of some vital information, such as that the story he has been waiting for is now at the top of his "in basket."

Stories may also be "queued"—that is, assigned a processing sequence—as for composition or for typesetting, or use of the on-line printer.

In newspapers, especially, the use of disc storage, coupled with video terminals, has led to the development of "all electronic newsrooms" which have "automated" (*i.e.*, eliminated) the "copy boy," and which provide far more expeditious handling of editorial matter, offering better *editorial control* of the copy flow and production processes.

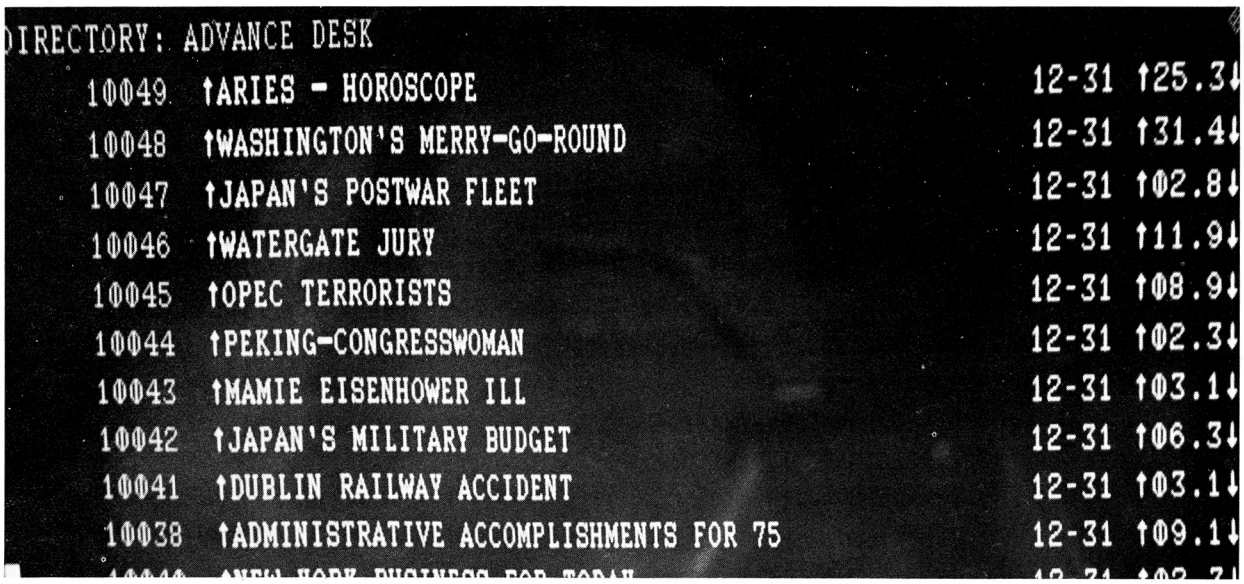


SLUG	ORIG		OPR		JOB	FROM	HJ	KEY
CONSL.01A	GALE	;07/09	GALE	;30/09	T	GALE	HJ 01430	0000672
JWS.001	LES	;30/09	JWS	;01/10		LES		0001430
SYS.TST	FLOYD	;30/09	FLOYD	;30/09	H	LES		0000031
COMM.01A	LES	;30/09				LES		0000376
MARK.UP	GAIL	;30/09	JWS	;30/09	hts	GAIL	HJ 05140	0000038
TERM.01A	GAIL	;30/09	OCRA	;08/09	hts	GAIL		0000000
SOUND.01A	VICKY	;30/09				VICKY		0000000
LOCKH.01A	VICKY	;30/09				VICKY		0000000
WILBF.01A	VICKY	;30/09				VICKY		0000000
INDIA.SPR	VICKY	;30/09				VICKY		0000000
SPHIL.01A	MARY	;30/09	MARY	;30/09		MARY		0000096
JWS	JWS	;30/09	JWS	;30/09				0000171

Commercial users of the Atex-8000 system often find uses for the queuing capabilities built into the system for its newspaper applications. This queue directory shows the job file name (slug), the originator or creator of the story, the date on which the job was created, the operator who last edited the job and the date on which this was done, the queue from which the job was forwarded; if the job has been through h&j, the total depth of the job will be shown, and, in this particular case, the number of keystrokes entered from the terminal will also be displayed. The actual format of the directory listings may vary somewhat from user to user.

Relevance to commercial applications

Similar procedures are also relevant to non-newspapers operations. For magazines, as well as newspapers, "editorial" or "front-end" systems may be developed, enabling writers to record their preliminary drafts directly onto computer files, to edit and update them, to compose them, and even to re-write them for purposes of copy fitting.



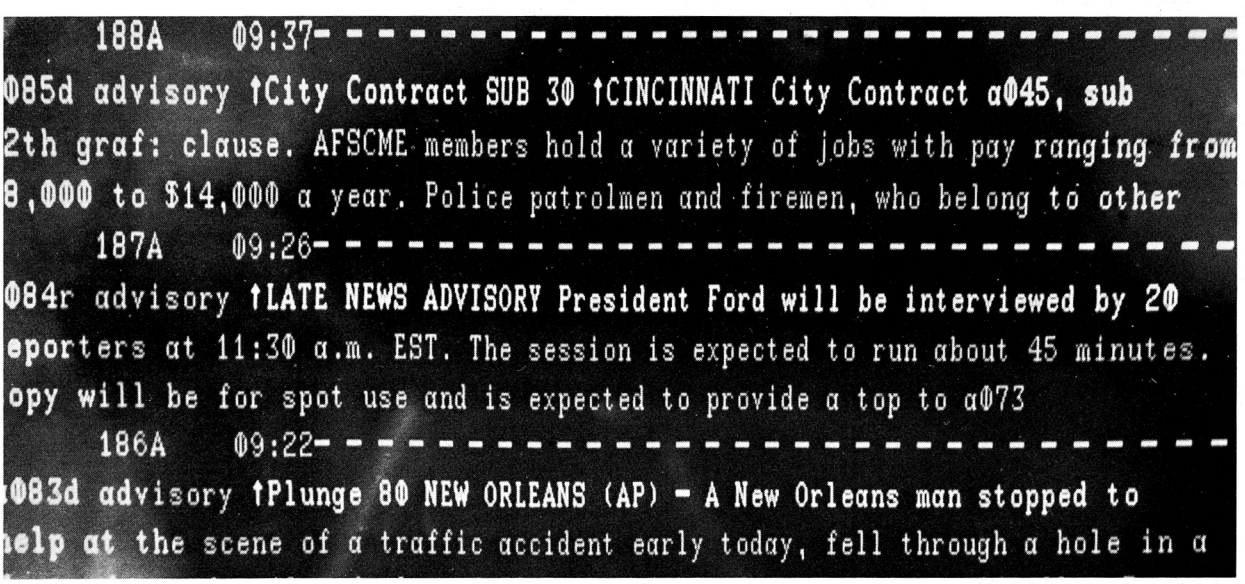
File directory: Short and long

The normal working directories in the Harris 2500 (newspaper) system are of the "short directory" format (above)—that is, a one-line entry for each story, identifying the story number, the slug line, and the depth of the story in inches. All items are listed in "last in, first out" order, with most recent entries at the top of the directory. The user may scroll through the directory and review the contents of his "in-basket."

Note that the listings are indented from the left-hand side of the screen. This gives the user space into which he can type any command he wishes to enter. He can fetch a story by typing "fetch" or simply "fe" in front of the story he wants. Similarly, he can transfer a story to another directory, print it, set it, etc.

Also note that the story depth (in inches) appears on the right-hand side of the screen. If the number displayed is of normal intensity, this is an estimate. If the number is bold, the story has already been through h&j and the depth count is exact.

Harris uses the long directory format (below) for wire service stories. This shows the same information found in the short directories plus the first three terminal lines of text. When a user requests a wire directory, he will get a listing of the most recent 96 items received over the wire. He may call to see more if he wishes.



In production shops, jobs may flow through the system (as from OCR input), according to designated job characteristics, and only those which raise questions or present problems may need to be "queued" for review on VDT's by operators who are skilled in diagnosing such difficulties.

And so the need to keep track of files, jobs or stories in order to process or revise them has opened the door to the development of techniques for the control of copy flow, the minimization of waiting time, the organization of like jobs for typesetting purposes, or the identification of problems, such as missing components (e.g., legends or tables) or jobs which have exceeded their assigned deadlines.

"On-lineness"

The most efficient method of operation is usually one where all of the functions and components are "on line"—from copy creation to typesetting, without the need to mount discs, paper tapes, magnetic tapes, film strips, or to feed in "parameters" of job specifications by manual means. Of course, such systems may be too expensive or too complex for smaller users. But as a general rule, the less the manual intervention the more productive the system—and hence the entire operation—will be.

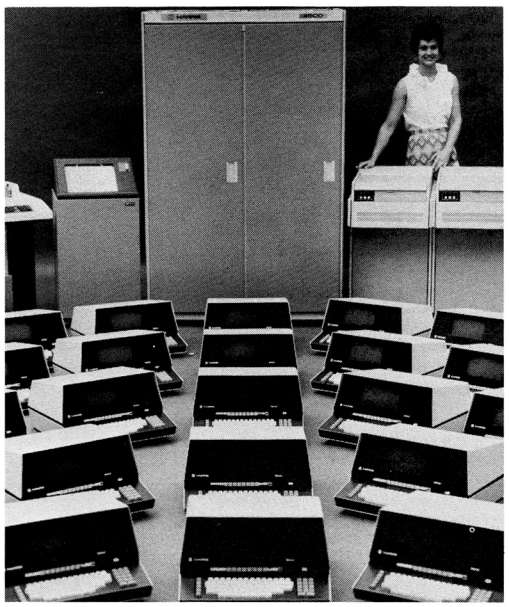
Definition of terms

We have referred to "editorial" systems, "front-end" systems and "production" systems. In all three instances we are assuming the use of video terminals and some file management capabilities, but for different purposes.

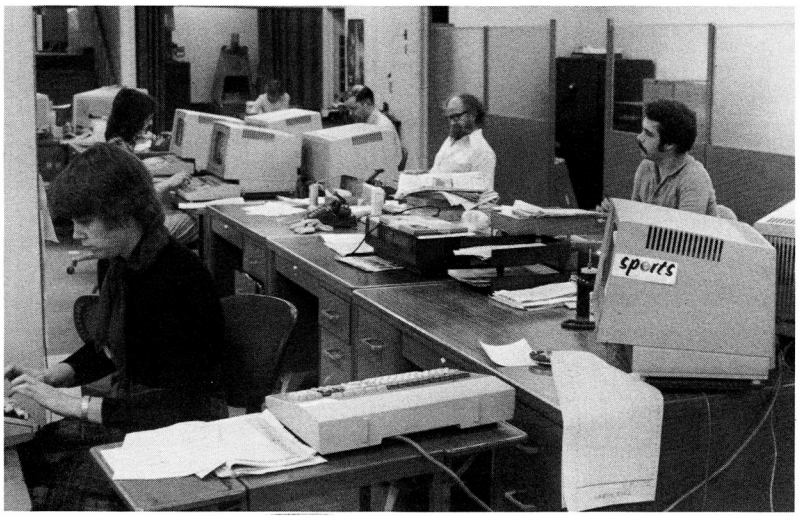
A "front-end" system is one which is used to capture writers' keystrokes so that their output does not have to be re-keyboarded and re-proofed to check for errors introduced in the input process. It is also one in which directories and queues permit the routing of material from one person to another and the subsequent review and revision of the initially-created documents. But if it is a "front-end" computer system, processing stops short of composition formatting.

An editorial system starts with a "front-end" capability, as above, but it also offers writers (or some writers) and editors access to the computer composition functions so that they can review the hyphenated and justified output and revise it for copy-fitting purposes. It implies, moreover, that they can revise the most recent version, as previously formatted, rather than to have to go back to the initial input. It also implies that it is editors (or some of them) who control the system, including the "production" into type of their material, by using appropriate commands from their VDT's. If page makeup is offered, it would also be editors who would determine and implement the page make-up functions.

A production system, on the other hand, implies that manuscript will be input in a production environment, that the re-keyed material will be reviewed subsequently by authors and editors who will mark their corrections on some kind of proof copy, and that these changes—even for copy fitting purposes—will be implemented by production personnel, probably using VDT's.



The Harris 2520 copy processing system, designed for use by newspapers with circulations from approximately 10,000 to 50,000. A typical 2520 system consists of four 1700 terminals, two 1520 terminals, a system controller, a 4.8 million character disc drive and associated peripheral devices.



Even in the smallest newspapers, video terminals are now being used by reporters and editors to capture, route and revise stories. This is the newsroom of the Independence (Missouri) *Examiner* (circ.: 20,000).

A back-end system is one which accepts unjustified input from a front-end system but performs composition and makeup functions without direct editorial intervention, although pursuant to editorial instructions.

A commercial typesetter offers a "production system" capability, but he may also seek to accept input from publishers, in which case he offers a "back-end" system, with the "front-end" activity being performed elsewhere. The publisher may or may not seek to provide machine-readable input with all necessary codes or commands properly included. The production shop may undertake to add, or modify, them, and it certainly would seek to "explode" them or to translate them into more system-oriented command language.

Review Questions:

1. What is the role of computer "core" memory with respect to text storage and manipulation functions?
2. What is the difference between "fixed head" and "moving head" discs?
3. How does "rigid disc" storage differ from magnetic tape storage?
4. Describe the manner in which the disc space is generally partitioned.
5. What is the physical relationship between a rigid disc and its read/write head(s)?
6. What is a disc "crash" and what are its implications?
7. Give an example of a "file management system" in which the user must assume at least part of the role of "file manager."
8. When a particular file is written in segments, on several different locations on a disc, how does the system know where to find each piece of the file?
9. What kind of information is contained in a file directory?
10. What is meant by a "cross directory search"?
11. What kind of an "audit trail" might be provided by a file directory?
12. When might "default" routines be applied to file management functions?
13. Define the following terms as they relate to file management:
    - Routing files
    - Messaging
    - In-basket
    - Queues
    - Directories
14. Delineate the differences between a "front-end" system, an "editorial" system, a "production" system and a "back-end" system.

20

Word Processing and its Relationship To the Composition Process

NOW THAT WE HAVE DISCUSSED editorial or front-end systems, and file management concepts, it must have occurred to the reader that there are certain similarities between office "word processing" systems and the input and revision capabilities which have been developed over the last few years for printing and publishing activities. This is indeed so, and it is difficult to draw a hard-and-fast line between a "front-end system" and a "word processing system."

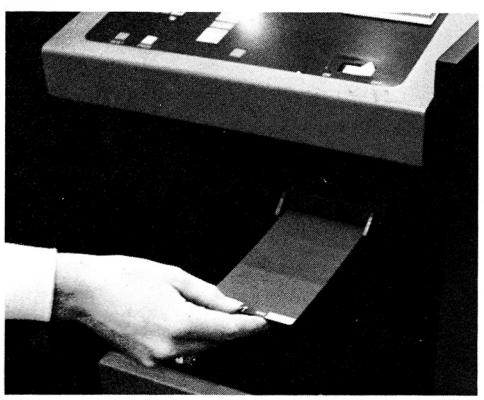
It may be that "word processing" implies control over the equipment by a secretary rather than by the writer or editor himself, but this seems to us to be too arbitrary a distinction to convey much meaning. Any front-end or even editorial system should seek to accommodate to the requirements and needs of the particular installation. Some writers *do* use typewriters and *can* use typewriter-like devices. Others *cannot* and *do not*. Some dictate, or "think best" when working with a pen and tablet, expecting a secretary to provide them with a subsequent clean script for further revisions.

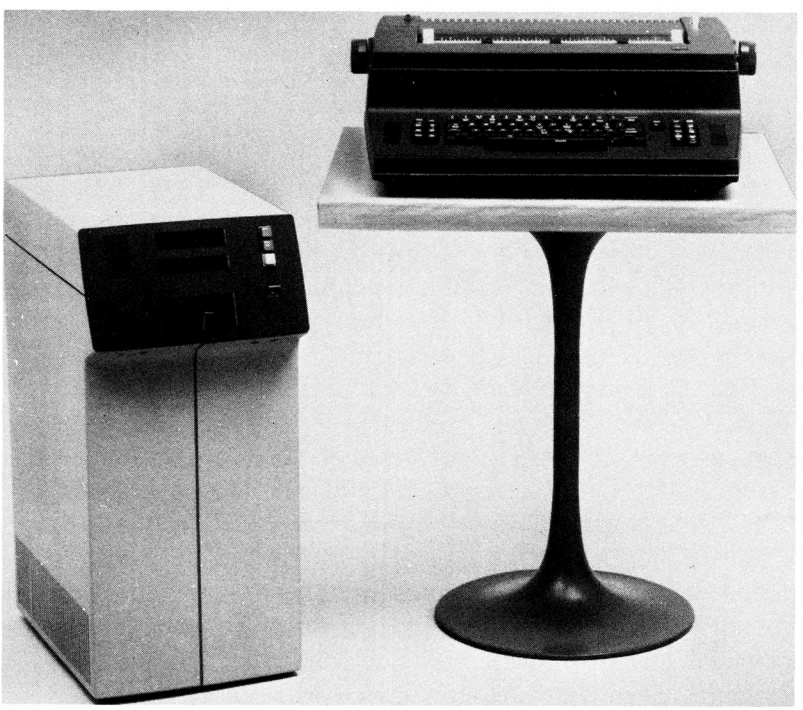
But at some point in the manuscript preparation stage—somewhere "upstream" in the "editorial process"—it becomes advantageous to capture keystrokes and to avoid redundant retyping by merging in corrections and revisions. The objective may be simply to provide the printer or typesetter with a clean manuscript, or it may be that the manuscript is not destined for typesetting at all. A shorter term objective may be to obtain "clean" copies in order to encourage further refinements in the manuscript and to secure the comments and contributions of others. But whatever the purpose, it is clearly facilitated by working—at some point of time—with a draft which can be modified and reproduced without the labor, cost, and time-consuming aspects of retyping.

Word processing equipment was intended to serve this purpose. It can also be served by the front-end or editorial systems we have been alluding to. The two applications are not mutually exclusive, but at this point in the evolution of office technology, word processing equipment is usually less sophisticated than the systems we have so far described. It usually does not offer directories and queues,

or control of copy flow. It is more likely to consist of "stand-alone" units or small cluster systems.

Originally, word processors were automated typewriters. Keystrokes were captured on paper tape or magnetic cartridges or cassettes. The material could be played back, the process interrupted, new material added, and a new version stored on tape. Hence a device for reading from tape and for writing to tape was required. To facilitate the retyping and correction procedure, it was (and is) possible to instruct the typewriter "controller"—usually, but not necessarily "hard-wired" and with limited intelligence—to search for certain "file marks," perhaps performing straight copying to that point, and then permitting the operator to proceed on a line-by-line, word-by-word, or character-by-character basis until the point has been reached where a change is desired. Sometimes the new version is written back over the old, or more usually onto a new tape, cartridge or magnetic card. There may be stringent limitations as to the amount of new material which can be "inserted," depending upon how the "buffer" within the typewriter is formatted. More recent versions provide a larger buffer within which material can be stored, and then written out, as to a mag card, but not to exceed the contents of the data which may be stored on the card.



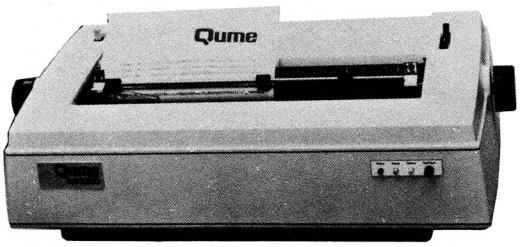


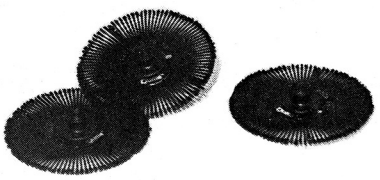
The IBM mag card usually contains a page of information, and it can be revised by re-inserting it in the control box which is cable-connected to the typewriter. *Right:* The IBM 6240 mag card typewriter, which operates at speeds of up to 55 characters per second utilizing a print wheel. *Above:* A mag card is being inserted in a Compugraphic WordCom reader which will transfer the information to a Compugraphic EditWriter.

While such devices are clearly improvements which offer great economies within an office environment, they are generally less powerful than the cursor editing available on a VDT screen, and the most recent developments in the word processing industry have tended to substitute the VDT and keyboard for the conventional typewriter, and a Qume or Diablo "Daisy wheel" printer for the typing element itself. Thus, the typing function becomes "output" while "input" and editing are performed on the VDT.

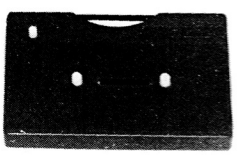


Use of magnetic cassettes allows greater editing leeway, particularly if two cassettes are used, as is the case with the Redactron word processing "system" pictured here.





Floppy discs have been added to some such systems to provide more interim storage, and there are even "shared intelligence" systems which provide directories to help locate the tracks on the floppy disc where text may be stored. "Shared logic" systems have been developed, with several users having the ability to access the same discs, or editing logic, or even character generating PROMs or ROMs. For offices needing more than one or two individual stations, shared logic systems are often cheaper—as well as better. The systems which are evolving look more and more like the "front-end" or editorial systems we have found to be so popular within the graphic arts industry. Word processing vendors are now offering "directories," "multi-tasking" and queuing functions, messaging, electronic mail, and many other functions which resemble the capabilities of "editorial" systems. They have moved into system-design architecture which is programmable, and they offer "sort" routines and other functions (such as column totalling or subtracting values from one column from those of another, to create a third set of figures, and even converting those into percentages.) Thus word processors are becoming more powerful and more generalized tools which are revolutionizing office procedures.



Although any IBM Selectric or other "golf ball"-type typewriter can be used as an input and output station in a word processor, the typewriter has come to be viewed more and more as an output device. The newer "daisy wheel" printers do not provide keyboards and replace the golf balls with flat, circular typing elements. These are capable of "escaping" by fine increments of spacing, and therefore of generating characters which appear to be proportionally spaced because of a reduction in their side bearings.

Because of the increasingly similar capabilities of both sets of vendors, several manufacturers of graphic arts equipment have sought to penetrate the word processing market—a daring move on their part because of the size and prestige of many of those already entrenched. And, in like manner, several of the word processing manufacturers and vendors have sought to provide systems relevant to the needs of the graphic arts industry. In fact, there are a few vendors whose attention is, at least at the moment, about equally dedicated to both markets.

But even before this rapprochement began, attention started to fucus on possible interfaces between word processing and typesetting, just as there was a great deal of interest in taking the cartridges used by IBM's "strike-on" composer—the MT/SC (Magnetic Tape Selectric Composer) or MT/ST (Magnetic



One of the earliest video terminal-oriented word processors, Lexitron's Videotype, still makes use of magnetic cassettes for text storage and editing, and goes to great lengths to simulate a typewriter. This VDT does not employ "cursor editing" techniques as such; instead, the operator rolls the text up and down on the screen by means of the platen, and spaces over to the point of correction.

Tape Selectric Typewriter) and trying to utilize the same stored text for typesetting purposes.

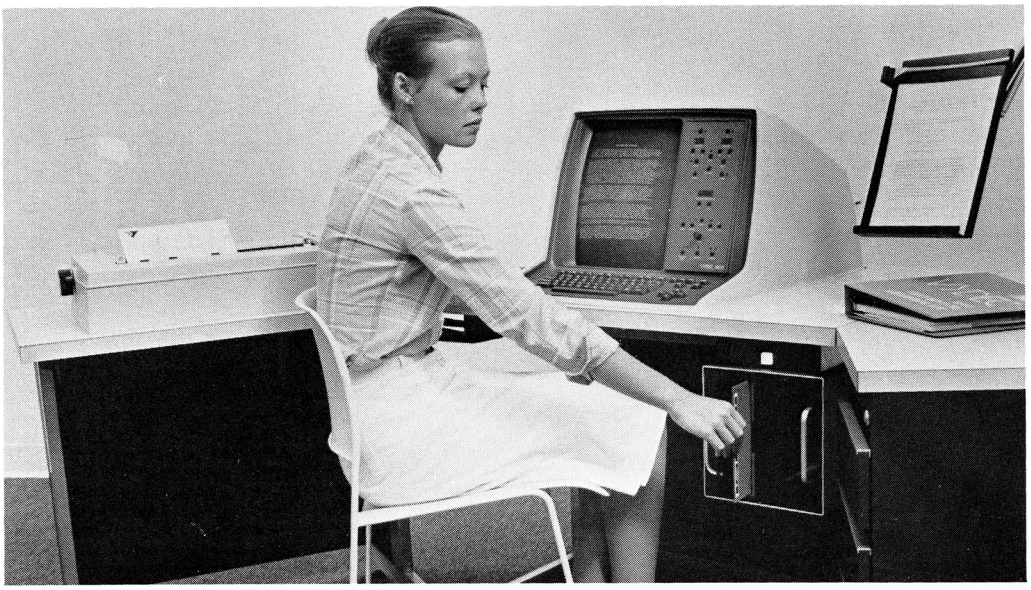
Three problems arise in this connection. One is a "hardware" problem of putting a device onto the typesetter or computer system which can "read" the word processing output—assuming that we are only interested in going one-way, that is, from the word processor to the typesetter or typesetting system. Or, conversely, putting a device on the word processor which can write out something which the typesetter or typesetting system can read.

The second problem is to translate into a code system which the typesetter expects, and the third problem is to incorporate into the translated data stream the typesetting commands which the typesetter or typesetting system requires.

These problems—especially the first two—are interrelated. For example, many of the word processors do not record their characters as six-, seven- or eight-level "bytes" written in a "parallel" fashion onto paper or magnetic tape. They are likely to record these codes serially. For example, if the word processor uses a Philips-type cassette such as you will find on the small audio tape recorders (although the signal is digital rather than analog), the coding will be "strung out" serially, and there will be extra bits written (sometimes) to signal the beginning or end of a particular character. There may also be some parity checking information. The writing process, moreover, will often not take place "incrementally"—that is, character by character—but it may take place by writing out a "block" of data at one time, with a gap between one block and the next, and so it is necessary for the "interface translator" to "know" how the bit stream is "blocked."

There is yet another hardware problem with the cassette approach, and that is the speed at which the cassette passes by its read or read/write head. For the recording of digital information there is usually no arrangement to pass the tape at a constant speed, and as the tape moves onto the take-up reel, it passes the write head at faster and faster speeds. Hence the recorded bits are farther and farther apart. Some devices simply "catch" these bits as they come along, and incorporate into the logic of the machine some rather elaborate error checking routines to make certain that all bits are properly read and interpreted. Other devices associate with the cassette tape, on a parallel track, a "clocking" or "timing" arrangement which indicates where bits are expected to be written or read from.

Even if "floppy disc" technology is used, rather than cassette, the "interfacing" problems persist. Floppies differ according to manufacturer. There are single density floppies, double density floppies, single-sided floppies, two-sided floppies, hard- and soft-sectored floppies, mini-floppies, and diskettes even from



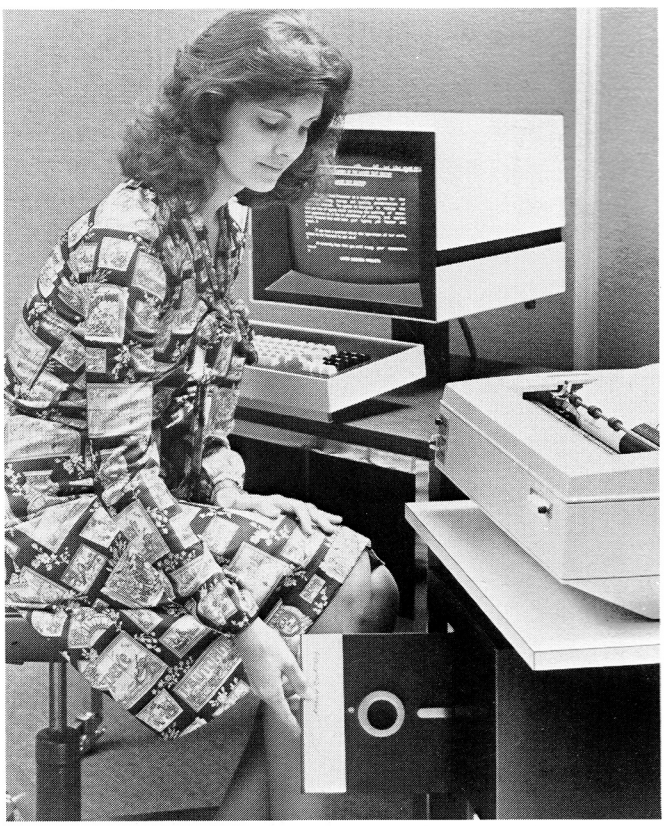
Vydec's Text Editors were among the first floppy disc based word processing systems. (This is Model 1400, dual floppy version.) However, the floppies are not utilized as true random access media. They are formatted serially, with each track on the disc corresponding to a page of typewritten material. The operator uses buttons on the terminal console to flip forward and backward through the disc tracks. Cursor editing and flexible formatting capabilities contributed to this system's popularity.

a given manufacturer may be encoded, blocked and formatted differently by the word processing vendor or the typesetter manufacturer.

It can be seen, then, that one of the very serious problems of devising an interface is the lack of uniformity so far as hardware is concerned. Often the best solution is to use the word processor itself to read its own data, rather than to seek to install a compatible reader on the typesetting system.

But after the data bits have been read, they must be checked for the identification of individual characters, the character coding must be interpreted and usually translated into the coding structure the typesetting system or machine expects to receive. This is the second problem we alluded to above.

The third problem is to get the desired typesetting codes into the text stream. Perhaps this is best done by reading the information through an editing terminal, where they can be inserted, but if the originator of the document will be willing to take the time and trouble to insert certain character combinations, such as "/t" for the text face, and "/p" for *end paragraph,* then some sort of translate can be provided. This may be done, in some cases, by the use of stored formats within the typesetter itself, or by computer program.



A more recently-developed device, the Lanier Text Editor, has a "transparent" file management system. Aside from performing the "insert," "delete" and "move" functions associated with cursor editing, many of these systems (including Lanier's) offer broader office-oriented capabilities, such as automated letter writing packages, automatic decimal alignment of tabular material & alphanumeric sort capability.

Not a casual undertaking

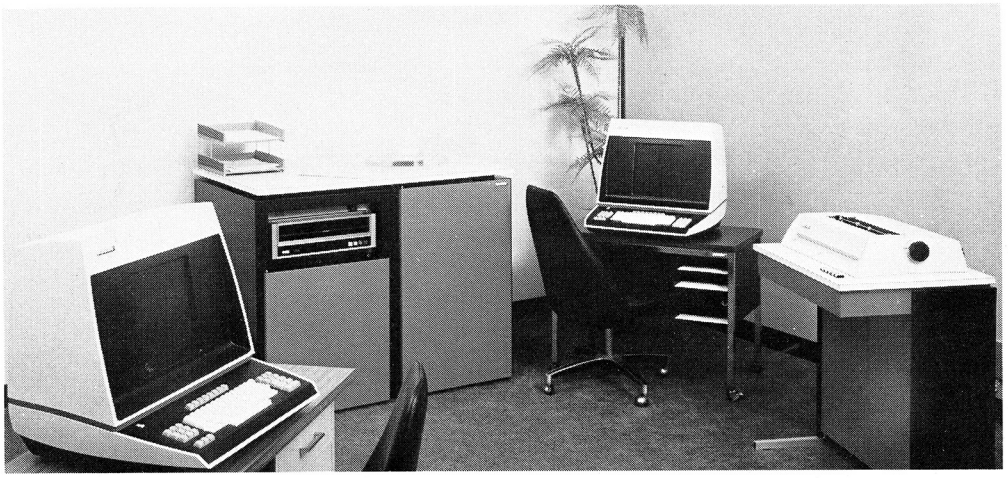
It should be obvious, then, that bridging the gap between word processing and typesetting is something that requires a clear identification of the kind of equipment to which one wishes to relate, how it works, what it produces, and what the receiving device (*i.e.*, the typesetter) needs and requires. Given enough time, and the expenditure of some money, this can be accomplished, and those who produce output via word processing may be trained to take on some of the burden of encoding their documents for the typesetting chore. These are not trivial tasks, however.

As time passes, word processing interfaces of a more generalized nature are emerging, with more programmable characteristics, but the hardware aspects of the problem persist, as well as the need for careful specification and understanding between those who create the data and those who take it further into the typesetting environment. Some of these problems are, of course, akin to those which arise with OCR copy preparation. And some practitioners have found it easier to "interface" their typesetting capabilities to accept word processing output merely by the expedient of suggesting that those with IBM Selectric word processor typing elements should prepare the final draft by the use of an OCR-scannable font.

For the future, we look for the evolution of better and more powerful word processing devices, which offer more editing power—similar, in other words, to the better devices now available for graphic arts' use. Ultimately we look for a complete fusion of the two technologies. This trend has, indeed, already begun. And the printout techniques which are being used threaten to supplant typesetting in certain markets or applications. High quality, proportionally-spaced printout devices are available, such as the Xerox 9700, the IBM 6670 or the Wang "Image" printer, using electrostatic or ink jet imaging techniques. More emphasis is thus being placed by word processing people upon hyphenation and justification and pagination. In the graphic arts industry, meanwhile, more attention is being focused upon bypassing the typesetting process, and going "direct-to-plate"—or, perhaps, using "printing-on-demand" approaches so that the process of publication itself may be affected.

The line editor approach

Another form of "word processing" is the use of time-shared text editing programs. These are available to in-house writers employed by firms or agencies with large computer installations, or to others who connect to commercial time-sharing networks by dial-up telephone circuits. Certain companies specialize in providing time-sharing text editing services. In some cases this is the major activity or function proferred by the network. In other cases, text editing is only one of many facilities made available to the user on his remote terminal. Some of the more popular text editing programs are ATMS (developed by IBM)—the initials stands for "Advanced Text Management System"—or WYLBUR, or T-Edit. Other proprietary systems include "Word-One" of Bowne Time Sharing, and similar offerings by Cyphernetics, General Electric, Beloit, and others.

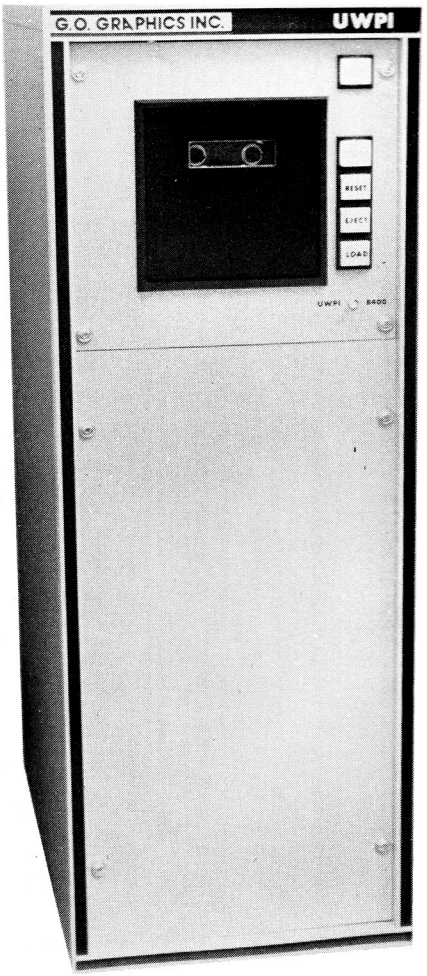


Left: Daconics/Xerox's Visual Type is a "shared logic" system built around a minicomputer and two 2.5-megabyte discs. Each controller can support up to 10 terminals and 10 printers. (Daconics uses gas plasma tubes, rather than cathode ray tubes.)

Below: Another type of system architecture, called "distributed intelligence," is used by Wang in its Word Processor 10, 20 and 30 systems. Microprocessor-controlled terminals can be configured as stand-alone systems with single or dual floppy discs, or as terminal clusters sharing a 10-megabyte disc.



Right: Conversion of material from word processing output to typesetter input may be handled in a number of ways. A few manufacturers of word processing equipment will provide an optional on-line typesetter interface or a means of generating TTS-coded paper tape, and some vendors of typesetting equipment sell magnetic card or cassette reader/converters. There are also independent firms which specialize in providing tools to bridge the gap between word processing and typesetting. The Universal Word Processing Interface from G.O. Graphics is such a product. It can be programmed to accept and convert magnetic cassettes from most common word processing devices into TTS code.



Most of these systems were devised to support typewriter-like devices, whether of the IBM Selectric variety, or the GE Terminet, or some similar machine equipped with a keyboard and a hard copy print-out unit. It is also possible to install video terminals, but these terminals usually behave as if they were typewriters, communicating with the system in much the same way, except that a limited amount of cursor editing may be possible—not by scrolling but by the use of overstrike or, perhaps, delete.

A "line editor" program usually works in the following fashion. You log on to the system, after dialing the number, getting a tone response, and placing your telephone into an acoustic coupler which is interfaced to your typewriter/terminal. You give your password, are welcomed to the system, and then you indicate what it is that you want to do, such as to create a new file. You are assisted by dialog or prompting, as your typewriter gives you instructions. Then you start typing, and each character is usually transmitted, keystroke for keystroke, although sometimes the transmission occurs a line at a time. If you make a mistake when typing, you can strike over the wrong character by means of "attn backspace," using the special key called "attention" as a signal.

After you have created your manuscript or some portion of it, you can ask to have it printed out on your typewriter with line numbers. Then, by the use of these numbers you can add or delete, or address locations within the line. The file, on the computer system, is structured in a line mode, and hence these programs are generally called "line editors." Lines can, of course, be reformatted, and rearranged. There are also paging programs which will terminate one page, generate a folio or running head, and start the next page, advancing the fan-fold typewriter paper at the appropriate place.

Our own opinion of most of these programs is that the dialog and response time involved interferes with the writing and editing process, and the cost, in terms of utilization, connect time, and performance, is often extravagant. Nevertheless, many large companies find it advantageous to support this activity on their in-house computers, and some small, isolated users find that the compute power which is made available to them at their remote location offers a better solution than to install a stand-alone word processor, especially if they wish to use the time-sharing service to perform additional functions. Usually the same terminals can access many programs, for data storage, manipulation, and computational purposes.

A fusion of technology

We are confident that we shall see the development of better VDT-oriented text editing programs for large, time-shared computer systems and minicomputer systems as well, coupled, perhaps, with the use of "distributed intelligence" at the terminal level. We are certain that the era of efficient text management is still unfolding. We look for a closer relationship and perhaps an ultimate synthesis of these three strands—the office word-processing approach, the graphic arts "editorial system" approach, and some successor to the time-shared line-editor approach.

Review Questions:

1. What is the primary function served by "word processing" equipment?
2. How do the cursor editing capabilites available on a video terminal compare with the editing capabilites of an automated typewriter with magnetic storage?
3. How are a word processing system's editing functions improved by the use of a random storage medium as opposed to a sequential one?
4. What are some of the factors which must be taken into consideration in converting word processing codes into phototypesetter-compatible language?
5. What type of hardware problems might be encountered in designing or adapting such an interface?
6. Do you think the use of floppy discs rather than magnetic cassettes or cards would ameliorate the conversion process?
7. How and when should typographic parameters be inserted in the word processing text stream?
8. How does the "line-editing" approach to word processing differ from either the "cursor-editing" or the "automated typewriting" approach?

21

Pagination By Computer And "Area Composition"

IN THIS CHAPTER, we shall be discussing not only page makeup, but also the construction or composition of "display ads" since there are certain similarities between the arrangement of text within an ad and the arrangement of text on a page. The term "area composition" is sometimes applied to both applications.

The dictionary definition of *pagination* has generally meant the process of assigning page numbers to a book, but by extension it also means the act of dividing a manuscript into pages. The term is used in the computer typesetting world to refer to the process of making up pages and outputting codes which, when they drive the typesetting unit, will cause the device to generate fully-composed pages.

There are a variety of tasks involved in the performance of computer-aided page makeup. These tasks naturally differ according to whether one is addressing the newspaper world, the magazine world, or the book world. They differ again if one is concerned with the arrangement of text and graphics in a display advertisement. Nevertheless, there are similarities between all of these assignments, and there can be some overlapping of hardware and software.

Why "computerize" page makeup?

Among the substantial costs of composition are those associated with putting type into pages. "Paste-up" of copy blocks is time-consuming, and it is not an easy task to do it precisely. If it can be automated, appreciable cost savings may frequently be realized. The quality of the product is also improved. Moreover, it will be possible to "go directly to film positive," or to film negative, or even, perhaps, to printing plate, bypassing film altogether, if one has the ability to compose material in pages in such fashion that it does not need to be altered, or arranged or modified by hand. Most newspaper production people are convinced that pagination is the logical next step, and, that it is on the verge of accomplishment. What is not so clear is how this capability will affect the flow of copy and the way in which paging decisions are implemented in a newspaper environment.

If page makeup can be made "interactive"—so that those who are involved in the process can look at provisional solutions, and alter not only the arrangement, the size and shape of text blocks, the measure, the type face and the positioning and scaling of the graphics, but even the content of the text matter as well—then there could be other significant advantages in terms of the timeliness and effectiveness of the product.

But there are many other applications where "interactivity" is not required or desired—at least not with respect to trial setting, or changing one's mind, or making last minute changes of basic material. Nevertheless it is not easy to position the relevant material in an attractive way because of such problems as associating footnotes with text, placing tables, charts and other graphics adjacent to the textual presentation, and organizing the whole so that it presents an attractive, consistent, and coherent appearance.

A bit of history

As of this writing, there is very little computer automated page makeup being performed except for the production of certain quite standard items such as telephone directory pages, or pages in the official airline guide, or for other "special-purpose" applications. For the most part, when it comes to photocomposition, type is composed in galleys, usually on photographic paper (so-called "stabilization" paper or resin-coated) and it is then cut apart and re-assembled on a carrier, with some spacing out where required, and some shifting of blocks of copy from one page to another to balance facing pages or achieve other desired effects.

Nevertheless, some typesetting companies had developed general-purpose book page makeup programs even as early as 1965 or 1966. The advent of high-speed photocomposing devices made the need for page makeup quite urgent since these typesetters could produce galleys at prodigious speeds—far faster than a crew of employees could arrange them into pages. Moreover, these machines sometimes had the capability of setting wide measure material, offering an opportunity to provide multi-column setting, and to avoid the stripping or paste-up that the arrangement of side-by-side columns would require.

One of the problems addressed, then, was the task of setting multi-column text. And the first efforts had to do with the arrangement of such text when it could be assumed that all "leading" would be of constant value. Even where heads were to be set in larger point sizes, the basic increment of "primary leading" would be maintained so that a line-counting program could be used, and if, say, 60 lines per column were cast off, one could assume that when three, four or five columns were arranged side by side, all text material would align.

There are two ways to achieve this. One is to compose a line within one column (say, line "1"), and then "tab" out to the beginning of the second column, where line "61" would be laid down, and thence to the third column, where line "121" would be composed. This was most readily accomplished on the ATF B-8



In the 1970s, most pages are "made up" by hand. In a newspaper environment, for example, a preliminary layout sketch is "roughed out" for each page. As the various elements (ads, stories, pictures) comprising each page are finalized, the layouts are revised. The final typeset galleys arrive in the composing room, along with the "final" layout. However, last minute changes often necessitate a quick conference with the layout editor.

Credit: *Inland Printer*, May 1976.

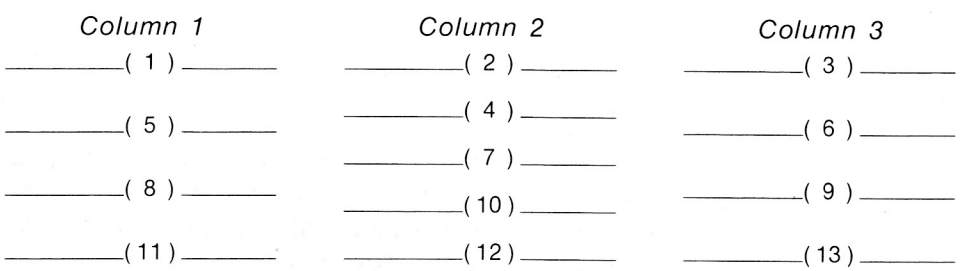


Credit: *Inland Printer*, May 1976.

typesetter because this typewriter-like device had tab stops which could be set manually just as on a typewriter. Unfortunately, few machines with higher speed capability could claim this convenient facility, and it was necessary to devise a more elaborate solution. This required that every line be justified by the computer (and not by the photo unit, unless it had a "multi-just" feature, which was seldom, if ever, present.) If all spacing elements were calculated by the computer program, and "put out" on the output tape, then the photo unit could simply lay down spacing increments along a line consisting, say, of three columns, with additional units of fixed space (perhaps two picas) in the column gutters. Of course, if one width value was short in the first column, the other two columns across that line would appear out of position.

This method of composing multi-column material is still used today by computer programs that set across the page. A "column merge" program is all that is required. Text is first arranged in galleys. Lines are numbered or otherwise identified, and then the first line, the 61st, and the 121st are arranged side by side by pulling them in an orderly fashion from magnetic tape, or, more likely, from disc. In the early days you might write out the first 60 lines to magnetic tape station #1, the next 60 to tape station #2, and the next 60 to tape station #3, and then pull off each line in order, providing for film advance after setting the righthand column.

An additional degree of sophistication is required, however, if the material does not align. If you are moving from 10 on 12 for text, to 8 on 9 for extract, then the program which arranges the sequence in which lines are to be set, and the amount of film advance, has to be quite a bit more complicated. You might set, for example, 10 on 12 text in column one, 8 on 9 in column two, and 10 on 12 in column three, for the first line across the page. Then you would have to drop down 9 points, move to the second column, set the second line of 8 on 9, and advance three points, moving back to the first column, and set the first and third columns 10 on 12. Then you would advance the film 6 points, set the next line in the second column, *etc.*



Note that for clarity the leading in this example has been increased to double the amount given in the text. Thus, the first column, which is given as 12 points of leading, actually has 24 points, etc. Numbers in parentheses refer to the order in which this material would be typeset.

But one no sooner grasps the solution to the challenge of multi-column setting than he is beset by complications he did not foresee. When he pulls his lines from the various tapes, he loses the precedence code information, which his typesetting unit requires. Consider, for example, the possibility that the first 55 lines of text are to be set in roman, and the next 20 or so in italic, and then the balance goes back to roman. Line one should be set in roman, therefore, but line 61 should be in italic, and line 121 should be back in roman. But the instruction to enter italic will not be encountered until far down the page. When it is encoun-

tered, then all three columns would be set in italic! Obviously, the practice of giving the typesetting machine an instruction to go into a type face or size and "stay there until I tell you to come out" will not work for this kind of setting without some modification. The program that chops the text into columns must also "put out" at the beginning of each column line what the photo unit needs to know about that line. This is what programmers sometimes call "lineinf" or "line information."

An alternate solution

But if the photo unit has "reverse lead," the problem of multi-column setting is greatly simplified. Each column can be set independently, for the required depth, then the photographic film or paper can be reversed, and the second column composed. The only "lineinf" required would be concerned with the displacement of the line across the page. A similar solution is possible when you output to a "full face" CRT typesetter, which does not need to reverse the film but simply its writing beam.

Problems of definition

Perhaps we were mistaken in plunging right into a rather complicated discussion of a particular problem of page makeup before providing some theoretical groundwork. However, the fact is that historically that is pretty much the way it happened—that is, one of the first pagination problems the industry tried to solve was multi-column setting, and the price it was prepared to pay at the time was to remain on uniform leading increments throughout the entire page. It was not long, however, before programmers and their mentors began to yearn for the ability to provide for more intricate solutions to the page makeup problem. Not all requirements were met by the simple capacity to lay down three or four columns across the page. In fact, one discovered that the business of composing even a single-column page was surprisingly complex. Gradually it became possible to identify many of the problems and to come up with acceptable solutions for some of them.

The not-so-merry widow

The most notorious problem is the *widow*. Programmers who had never before read a book, let alone appreciated its typographic niceties, suddenly learned, to their consternation, that to begin a page with a widow was not *comme il faut*. Nobody was entirely clear where this convention originated. Theirs not to reason why! Someone hazarded the explanation that if the first line on the page was short of the defined measure, as it would be in the case of most paragraph endings, then the page did not have a proper "frame." However, if the line at the top of the page consisted of a complete paragraph, which was only a partial line ("Here she comes now," he cried.), somehow the "frame" argument was set aside.

It is not our privilege, the computer typesetters were told, to do away with these established conventions. Our challenge is to achieve the same solutions.

Find out what it is that the hand makeup man does. Once you know what goes through his head when he divides galleys into pages, then you can write programs to do the same thing.

A "widow" can therefore be defined as a quad-left line that is immediately preceded by a justified line. If the preceding line was also a quadded line, then the line under consideration must be a stand-alone line which need not necessarily be rejected from a position at the top of the page.

Now that we can recognize the widow, we can avoid placing her at the top of a page or a column. Unfortunately, we must then deal with the problem of having our pages or columns come out with unequal depths.

What is the rule about page depth? we asked. We learned that in the United States it is permissible for pages to be "a line long or a line short" so long as facing pages balanced, and so long as facing pages which were short were not followed by facing pages which were long (or *vice versa*). In Great Britain, oddly enough, it seemed that all pages, except chapter endings, had to come out precisely the same. When we asked an old-line typographer how this was achieved, he said he couldn't rightly remember. It was jolly well done, but it had been so long since he had had to perform the operation himself that he had jolly well forgetten. He thought for a while longer and then recalled that it must have been done by resetting some of the bloody lines, making them tighter or looser so that the "little lady" (the widow) got swallowed up. At that point the olive in his American martini also got swallowed up and he lost interest in the subject at hand.

It turns out, however, that precise page depths are often achievable if there are elements of vertical spacing in the page. An odd amount of space separates text from extract and extract from text. This can be stretched or shrunk a bit. There is room for a little "fiddle" (an anglicism again!) above and below the heads or around illustrations. Usually this does the trick. But woe unto you if these elements are not present to enable you to perform "vertical justification."

That term became very popular, for one suddenly saw that the problem of coming out to a fixed depth vertically was precisely the same kind of problem as coming out to a given width horizontally. There are fixed spaces (the lines themselves) and justifying spaces (the extra white space). And just as compositors sometimes "cheated" by indulging in letterspacing (space between the letters) so they were tempted to cheat by "carding"—adding strips between metal type slugs so that forty-one lines took up as much space as forty-two, and the final line was pushed over the top of the next page, to drive the widow down.

Could the photo unit "card" discreetly? Some could and some could not. Increments of an additional half point between some lines (starting down from the top or up from the bottom) provide too crass a differentiation, even if the typesetting machine could advance its photographic film only in such increments. Later, machines came along that could do this in tenths of a point, which means that a deviation between two lines of this amount can hardly be observed.

"Carding" is not a sufficient (if acceptable) answer, however. Some pages simply can't be carded out, and even if there is additional white space for vertical justification, too much space might be required to bring the page to the proper depth. Why does this problem come to light? It is because there are certain situations, within the text stream, where it is not permissible to break the page even in the absence of a widow. What are some of these conditions?

When a head or subhead occurs, it is customary (indeed, it would look odd otherwise!) for two or more lines of text to follow before the page can be broken. If a table appears, the table must be kept as one unit and not split over two pages. If the author waxes poetic and makes his point by a stanza of verse:

> Our author would surely explode in a rage
> If *this* line broke to the following page!

One assignment we encountered fairly early in the game was a listing of lawyers, with their titles, birthplaces, and legal pedigrees:

> Jonah W. Jones
> Senior Partner
> Hickory Rock, Pa. 1878
> Admitted to practice before the Supreme Court of Pennsylvania in 1905.

It was not permissible to disassociate Mr. Jones from his title, nor from his birthplace. You could put his legal pedigree on the following column on the same page, or on the following facing page, but not on the obverse side of the same sheet.

It becomes evident, therefore, that there are certain "no-break" areas of composition. Two problems arise: one is to recognize these areas (how to tell a "head" from text or a rhymed couplet from parlor conversation), and the other is to make pages come out even despite the lumpiness of the elements one has to contend with.

Tables and illustrations have to be recognized as coming within one of two classifications: either the tables (or the white space for illustrations) has to be positioned precisely where the text requires them to be (in which case they become "non-floatable no-break areas") or else they can be "floated" to another position if they will not fit directly beneath the text that references them.

Footnote callout references and the accompanying footnotes also pose very serious problems. If the reference occurs on the page, generally speaking the footnote must also be placed on that same page. And there are times when the footnote is so long that without it (and the line which references it) the page will be impossibly short. With both elements, the page becomes too long, and the footnote, which could be even larger than the entire allotted page size, must be carried over to the bottom of another page.

Our discussion of these difficulties, which began with references in the past tense, is not intended to suggest that these matters have now all been solved. Far from it. But the major decisions have now been identified, and some program solutions have been developed.

Conceptual solutions

The problem of book page makeup can be approached in a number of different ways. Perhaps the most ambitious solution is what might be called "automatic page makeup" wherein the computer program presumably solves all pagination problems, and produces acceptable results without intervention or further editing.[1] A second approach would be one in which the computer arranges solutions where it can, and identifies problem areas where a solution cannot be determined on the basis of the parameters which the system has specified. The user can decide whether to deviate from those parameters, modify the text, or provide an answer which the program was not "bright" enough to discover. (For example, by backing up to an early point, which is beyond the programmatic purview of the software.)

1. The RCA "Page 1" program is illustrative of this approach.

A third approach would be an automated "cast off" program, wherein the program suggests or indicates page breaks, but leaves it up to the user to accept or reject them. Presumably this would be done on a page-by-page or column-by-column basis, with the user scrolling the text material on a video tube and inspecting the locations where the program suggests that pages be terminated.

A fourth approach would be an automated "paste-up" program. Here the user would tell the computer precisely what he wants on each page and in what position, and the program would merely translate those instructions into output commands to the typesetter.

In all of these cases, the software would be armed with a "vertical justification" capability, so that it would space out or in to the extent allowed, once it, or the user, had determined the page or column break locations.

Totally automated solutions

If a satisfactory solution can be totally automated, there is no need for user interaction, and the program can be run in a batch mode. For certain kinds of material this is a satisfactory way to proceed, and in these cases it may be possible to accept text which has been carefully edited, and run it through typesetting and page makeup without further review. Certain work does indeed lend itself to this approach. And within this category there seem to be two alternate ways to proceed. One is to take the page depth as given and invariable, and to pour text into that page, regarding it as an area, and attempting to fill it up while hyphenating and justifying. A *trial setting* mode can be specified, and if a widow should appear at the top of the next page, or a copy block emerges which cannot be broken, then the program will make an effort to resolve the problem first by ver-

tical justification, including carding, if that is permissible, and second by re-hyphenating and rejustifying, if necessary over and over again, until a fortuitous solution is determined.

An alternative way to proceed is to assume that all lines which have been hyphenated and justified are—in effect—slugs of type. They are not to be reset. The problem is to array them logically. But if problems arise which cannot be resolved within a defined area or boundary, then the boundaries can be altered on facing pages—by going long or short. This approach implies that the program does not need to know what the text is all about. All it needs to work with is the facts about the text, and it can manipulate these statistics to try to get the best possible solution not on a page-by-page basis, but on a chapter-by-chapter basis, again shuffling back and forth the obstinate elements, spacing in or out wherever allowed. Perhaps some combination of these two concepts is also possible.

Totally automated page makeup programs were popular in the days when batch processing was the rule, and interactive intervention by means of video terminals was not available. As we have indicated, there is still some significant need for such programs, especially for relatively routine assignments. But one difficulty which arises is that if the program cannot in fact resolve the problem posed at a particular point in the manuscript, some change has to be introduced and at least the balance of the job—if not the entire job—must be re-run. And it may be that the second time the program is only able to advance another page or so before it again runs into a problem area. Consequently, some degree of interaction is clearly indicated, whether on a page-by-page basis or on an exception basis.

At this point it is necessary to determine whether the person who then makes the paging decisions is given the discretion to modify the text to fit the space, or must work with the space to accommodate text which may not, in fact, be altered. (One could not re-write a verse in the Bible to avoid a widow, but many living authors would not object if a modification were made here and there in the interest of copy fitting.) Consequently, in the "ultimate" interactive system one usually presumes that there should be the capability of addressing, editing and reprocessing some portion of the text as well as merely spacing it, or re-processing it without editing.

Embedding make-up parameters

Typographic commands are at present sufficiently powerful that some elements of make-up (less ambitious than the "totally automated" schemes) can be incorporated into the text stream. You can, for example, insert a "mark" so that later you can issue a command "reverse to mark" to lay down a second column, properly indented. There are even commands to permit you to cause a certain block of text to be divided into three columns of equal length. But the problem still usually exists of how to visualize what it is that you wish to achieve. Here is where a "soft copy display" or an interactive display device is especially helpful. If, on the other hand, copy is written (and rewritten) to fit a designated space—as with a news magazine or many catalogs) an interactive solution may not be required.

The use of the video tube

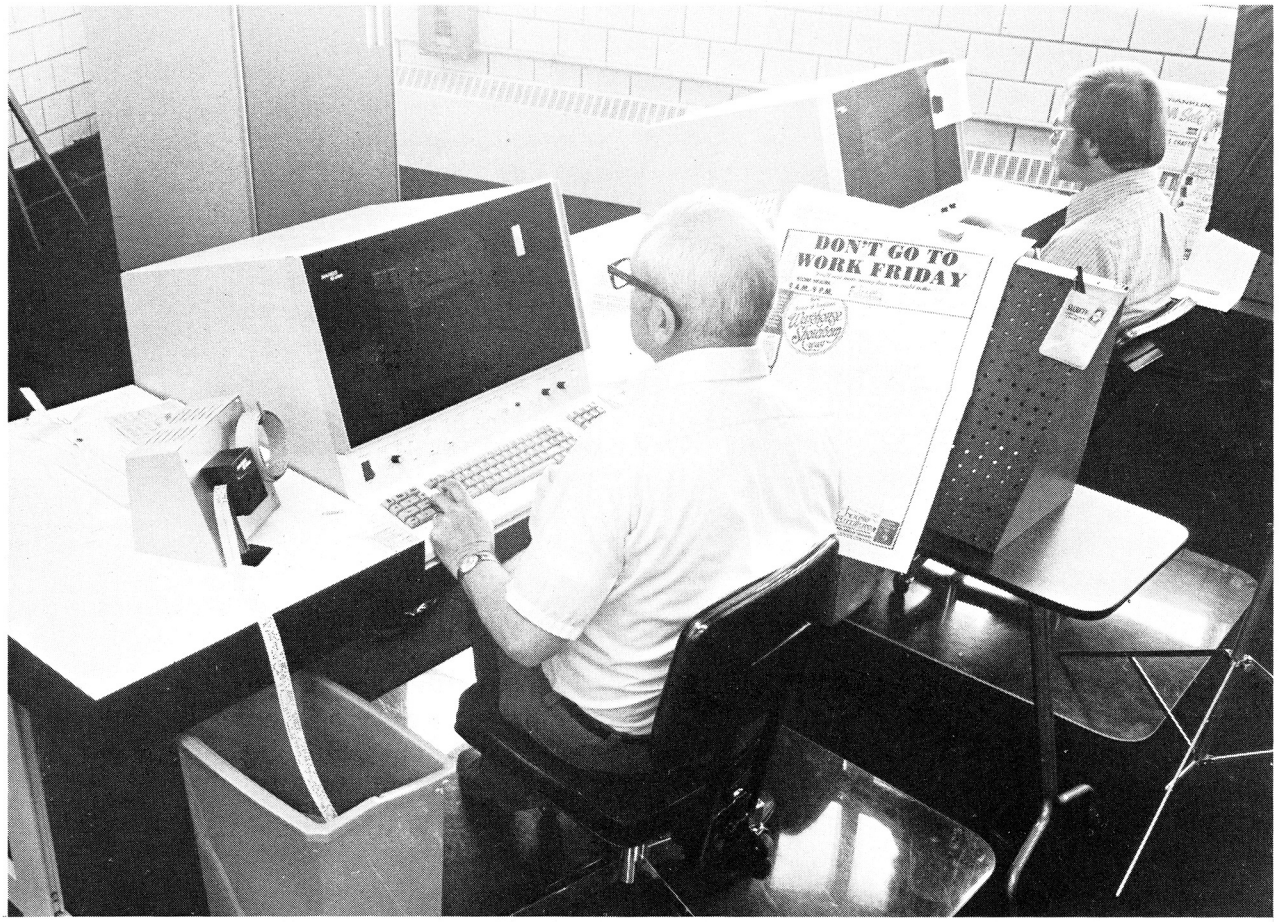
Turning, then, to interactive page makeup possibilities, making use of video display technology, we find that the first significant work in this area had to do with display ad layout rather than page makeup. Perhaps this is because the task may be simpler. But more likely it is because the most promising market appeared to be with newspapers, and newspapers were not then ready for full-page composition, lacking, at the time, systems to handle news material with sufficient speed and efficiency. On the other hand, they *were* ready for some improvement in the method of composing display ads. Now they are ready for both—or nearly so.

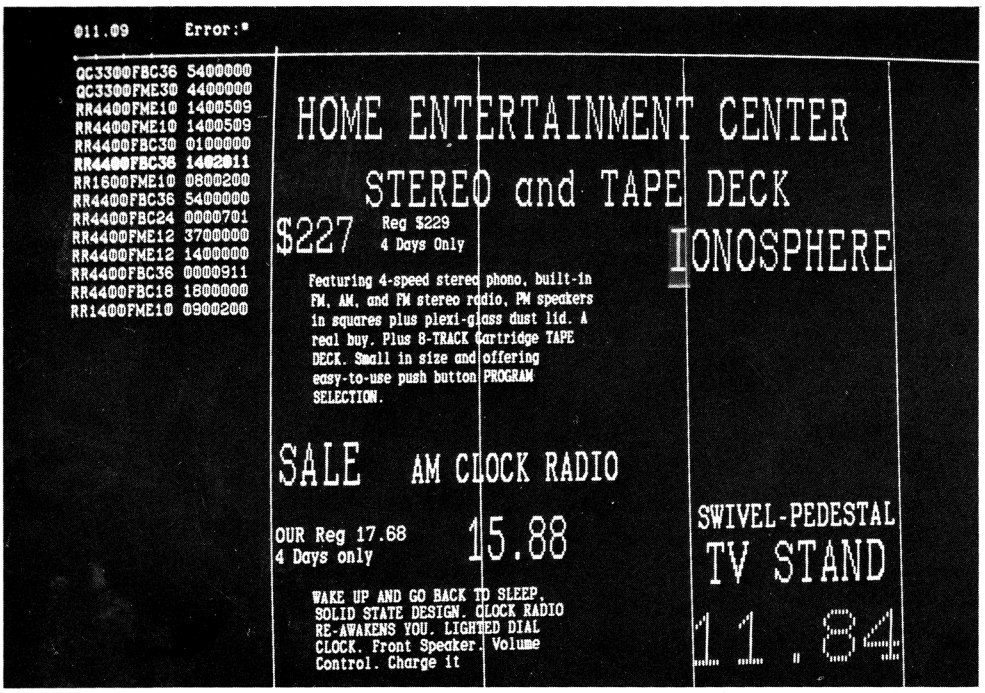
The first interactive solution was again proposed by the Harris Corporation. This time it was the Harris 2200 ad layout terminal, and for several years Harris occupied a unique position as the only vendor of a device with functional characteristics sufficient to handle the problem of display ad composition.

The 2200 accepts input from paper tape keyed outside the "system" or it will now accept it from input which has been stored on disc, from a 2500 "editorial" system or that of some other vendor. The input consists of the type elements which will go to make up the ad. These need not be marked up as to type face, line length, size, or any other characteristic, so that the person doing the initial keyboarding does not require any knowledge of layout or typesetting coding conventions. The 2200 operator reads in this text, which is arrayed on the right-hand side of the screen, with arbitrary type face parameters assigned on the left-hand side. He selects an element and assigns to it the desired typographic qualities. This he can do in two ways. He can change the typesetting parameters which appear on the left of the screen by moving his cursor to that area and overstriking the information which is there with other data—such as overstriking type face 1 with type face 3, and size 10 with size 18, and line length 20 picas with line length 14 picas. As he makes these changes the material on the right-hand side of the screen takes on the characteristics he has indicated. The block assumes the size and shape he has described. The material is counted for line length and, if necessary, is hyphenated. He proceeds to do the same thing for all the elements, as he visualizes the ad, or follows the supplied customer layout.

But the operator also has available another pad of keys which permits him to "bump up" the size of the type, or to "bump" it down, one available size at a time, and to see how it fits and looks under these circumstances. He can also move the block of type left or right, up or down, and the parameters on the left of the screen are altered accordingly. When he is satisfied with the appearance of the ad, he can pass it on to the output typesetter and it will be composed just as he formatted it.

Thus the operator can get what he sees, subject to certain limitations. He does not see the actual shape of his characters, but an abstract representation of the average shape of a character, which does not have the weight or appearance of the type he is selecting. And he does not see the space that character takes up,





The Harris 2200

The Harris 2200 was the first display ad composition terminal to be introduced and marketed. It is used primarily among newspapers although some commercial typesetters have also installed 2200's for ad work.

Above: Harris 2200 display ad composition terminals in use in a production environment.

Left: Ad copy on the screen of the Harris 2200.

but he does see precisely the amount of space that the line or block takes up. It is "stretched" vertically and horizontally so that it fills the space the "true" type will in fact fill up.

Another limitation is that, in moving one block of copy, other blocks are not automatically repositioned, unless they are linked together in some hierarchical arrangement. Perhaps this is as it should be, because the program does not know what belongs together and what should be manipulated separately. But once the operator has linked certain blocks into a certain relationship the entire structure of related blocks can be moved about the screen as an entity.

Finally, there is a limitation as to the size of the characters which can be displayed and the size of the ad which can be composed. Harris has overcome this limitation somewhat in its newer 2250 model, which permits some reduction or enlarging of the image (from "true size") and some sidewise scrolling to represent material which otherwise would exceed the width of the display.

The point that needs to be made is that for the composition of display ads—even when the function is not to "create" the ad in the first place, but to reproduce what the original sketch intended, with whatever degree of precision the artist or agency layout man specified—the operator apparently needs to see a fairly precise representation of the type to be composed.

Camex and Raytheon

Two other vendors subsequently developed interactive ad layout and mark-up terminals. They make use not only of a keyboard but also of a tablet, with which the operator can draw shapes and contours within which text can be "poured"—and hyphenated and justified to fit. In addition, Raytheon offers a "menu" of certain standard shapes—elipses, circles, squares and the like, to assist the mark-up man in the composition of text to certain elaborate boundaries.

Both Camex and Raytheon go beyond Harris in that they do in fact come closer to representing the actual position of the individual character as well as the general size and shape of the block. Their terminals are also perhaps more interactive in the way in which it is possible to move material from one place to another on the screen, and to link or de-couple blocks of text. It appears to be easier to locate type with reference to the positioning of graphic elements with these two products than is the case with Harris, since the 2200 and the 2250 do not set type to contours or permit the outline representation of artwork on the screen.

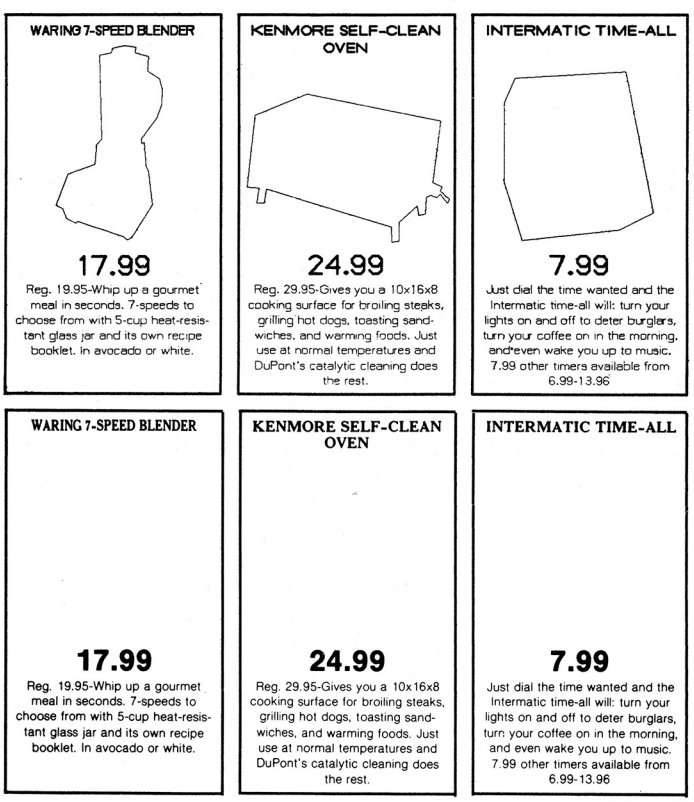
The Camex technology was subsequently made available to Compugraphic, and a low-cost, micro-computer version, operating much along the same lines, was developed. A British Company called Xenotron developed an approach much like that of the Harris 2200, but using a "area cursor" which could be adjusted in size and position prior to the recomposition of the actual text. No doubt there will be other low-cost products, which may or may not have all the same capabilities.



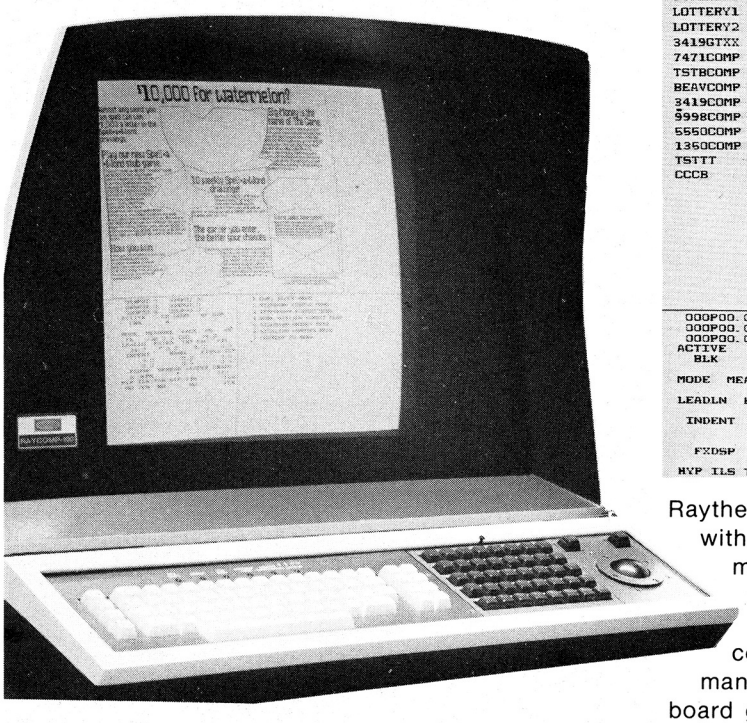
The Camex 135 system

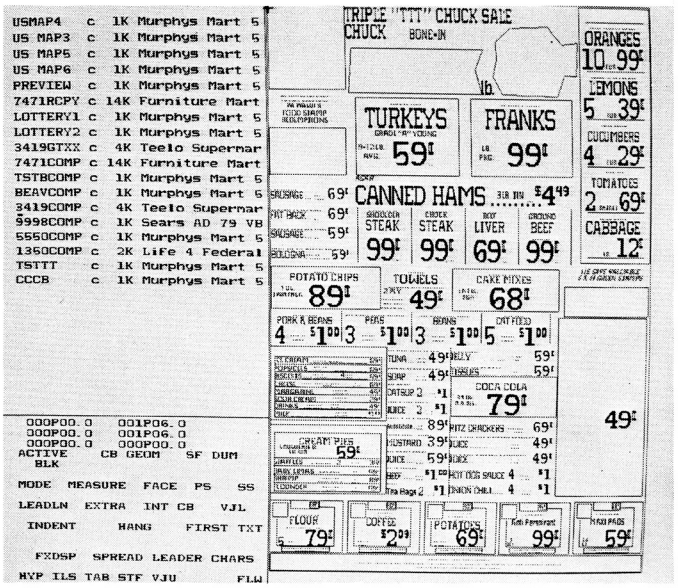
The Camex display ad composition system uses a random-stroking cathode ray tube rather than the more commonly used raster scan. Another unique feature of this system is the fact that the graphic tablet is used as the primary means of communicating with the system. In the picture above, the operator is "picking" the composition parameters from a pickboard area which is superimposed on the graphic tablet. The keyboard is tucked under the display.

Left: Proof printer output (top row) and typeset output (below it) of the ads composed on the 135 terminal. Note the close match between proof copy and typeset copy. The proof printer used by Camex is a Versatec which prints in a 200-dot-per-inch matrix pattern. The typeset output was generated on an APS-4.



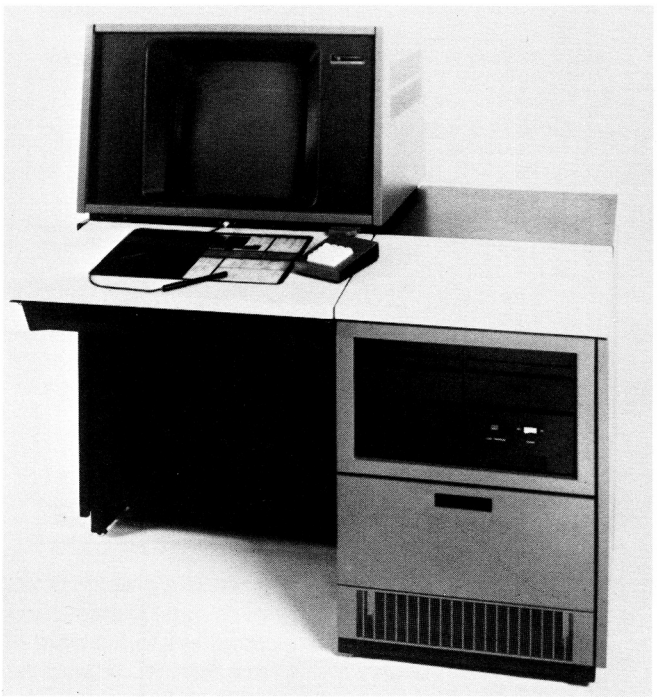
Raytheon's Raycomp-100





Raytheon's Raycomp-100 also makes use of a graphic tablet with a small pickboard area. However, most system commands are issued via the function keys to the right of the main keyboard. There are a number of template overlays, each related to a different phase in the composition cycle, which are used to expand the command repertoire. The trackball at the far right of the keyboard governs the movement of elements on the display. *Below:* A full-page grocery ad being made up at the terminal.



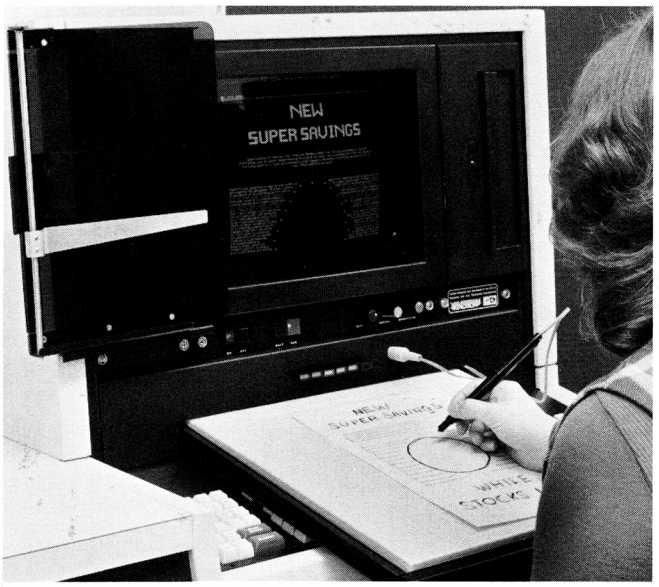


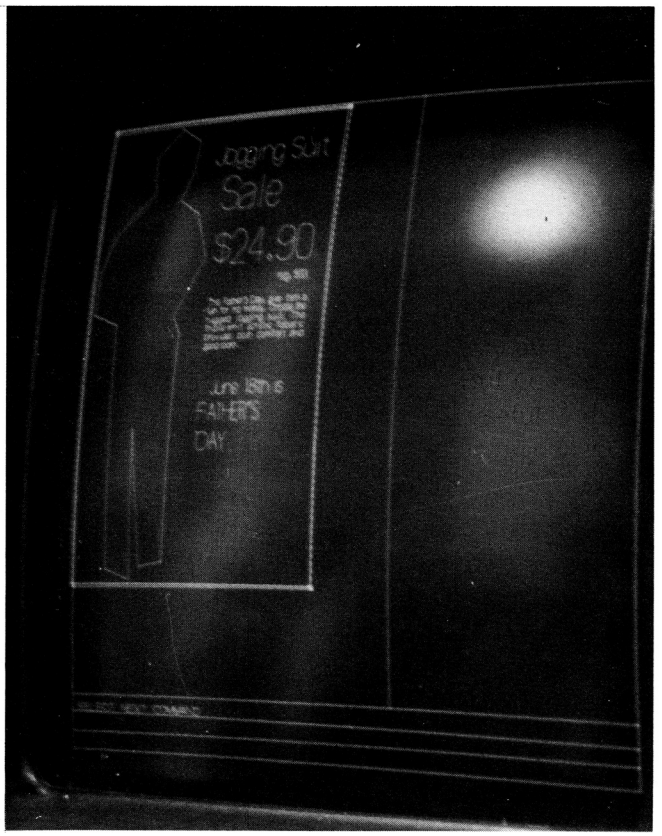
Compugraphic's AdVantage

The Compugraphic AdVantage (above) was first introduced in the spring of 1978, and first delivered to customers in the spring of 1979. It is a microcomputer adaptation of the Camex 135, by agreement with the latter company.

The screen (above right) shows outlines of illustrations, around which text can be positioned.

The AdVantage, like the Camex 135, makes extensive use of a pickboard (right).







Left: Xenotron has added a graphic tablet to its Video Composer, and now also offers a horizontal and vertical scrolling buffer capable of displaying a full-sized broadsheet page.

It is significant to note that Raytheon's device—called the Raycomp-100, is visualized as the first step toward a page layout terminal, and the same is true with Camex. Harris, on the other hand, evidently has different plans for the development of a page layout terminal. Both the Xenotron and the Advantage products appear to lend themselves to broader uses, as for magazine and brochure make-up, or perhaps even to serve as a device to which "exception" pages from a batch-oriented system could be routed.

Interactive page layout

It appears that when it comes to ad layout and mark-up, a fairly precise representation of the ad is sought to assure that it will in fact achieve the effects desired, and these effects are generally specified by the customer, and are not created by the operator. But when it comes to page layout—of newspaper or perhaps even magazine pages—it seems possible for the operator to work with a higher level of abstraction. He does not need to see the type—or all of it in all sizes—but merely some indication of what items are to be positioned at various locations and how much space they take up. Cruder stick images of type for headlines, and straight or squiggly lines to represent text, may be sufficient. On the other hand, the layout person would like the capability to "zoom in" on certain features and to have these lines magically converted into type characters so that he can read the text, and even edit it, if necessary, so that it will fit.

"Soft copy" displays versus interactive displays

One approach which has been used since the mid-1970s is the "soft copy typesetter." This consists of a visual display which simulates a typeset page or some other typographic element which needs to be critically examined, such as a display ad. The display can consist of a representation of that typographic element on a storage tube—such as a Tektronix 4014—or a refreshing raster scan TV monitor. Another method whereby such an image can be created is a "flat glass" plasma display device, or a random stroking VDT. In most cases the representations are not precise: they do not show the actual type, or perhaps even the actual weights and styles of type, but the approximate sizes of the individual characters is evident, although the x-heights and ascender/descender relationships are only approximated. At least two vendors have been able to show the actual characters which would be typeset, however. By diverting the image (or, more precisely, the character stroking signals) from a CRT typesetter to a video storage tube, one can then see precisely what the effect would be if it were typeset.

But such a soft copy display, convenient as it may be to help you visualize the graphic appearance before typesetting, is not an interactive solution. If you are not satisfied with the results you must correct the typesetting parameters by calling the file onto a conventional VDT for editing.

More useful, therefore, are interactive, parameter-generating terminals which permit you to compose copy, move it around on the screen, recompose it if

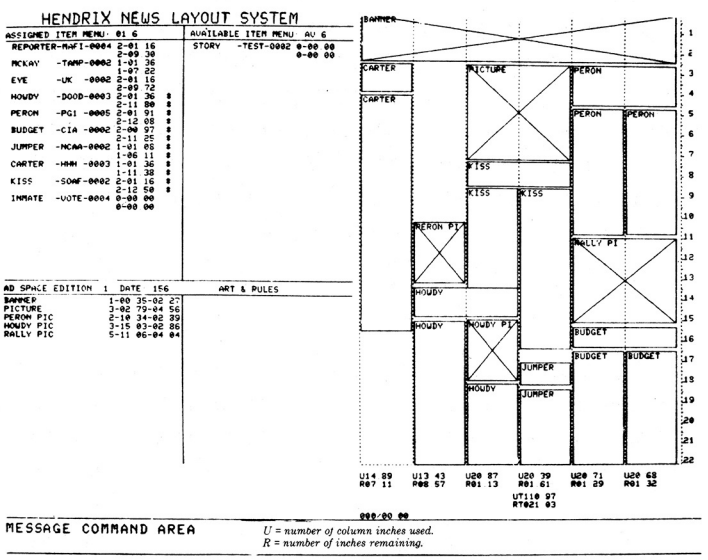
necessary, and then release it for typesetting. The display ad terminals we have already discussed are devices of this kind since, as you reposition the copy, or change its boundaries and the size and fonts of type, these parameters are incorporated into the text stream, or otherwise directly associated with it. In other words, you are creating what you see, and not merely displaying it.

An in-between solution is one which will permit you to move blocks of copy about on a screen, but not to recompose it. Such a device might be used for "automated paste-up." You call up the composed text and decide how much of it will fit on the page, indicating where columns should break, and where various blocks of copy should sit. You do this directly at the terminal display device, and it will record the positioned parameters which will then obtain. But you cannot edit or recompose the copy itself. Such a solution may be quite useful and perhaps sufficient for various kinds of make-up problems, especially if you can incorporate a command which will permit you to stretch or shrink the composed text in a vertical direction by means of "vertical justification." Atex has supplied the U.S.. Government Printing Office with terminals of this kind.

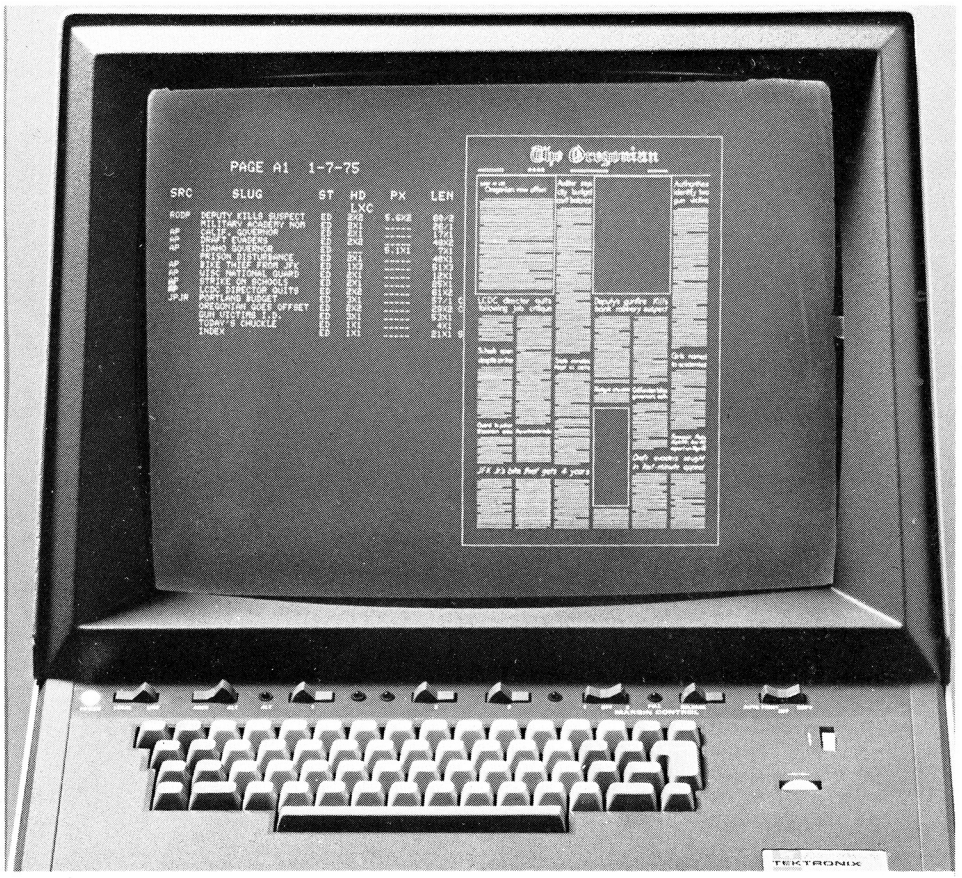
At the time we wrote our first previous "text"—*The Primer for Computer Composition*—in 1971, we said, "At the moment we are not aware of any programs for English language setting which provide an interactive and re-entrant capability for page makeup, or even for hyphenation and justification. A terminal-oriented program, which permits the user to talk to the computer in 'real time' so that the computer responds instantly to his commands, requires different programming concepts from those of the 'batch processing variety'."

We went on to say that "It is in the area of pagination, especially of the interactive variety, that we see the most challenging problems ahead of us. The only comparable area, also exciting to contemplate, lies in the extension of the use of the computer as a text editing tool.[1]"

1. John W. Seybold, *The Primer for Computer Composition*, p. 136.



As of this writing, several companies are experimenting with different approaches to interactive full-page make-up for newspaper applications. Hendrix has chosen to display a diagrammatic representation of the page dummy (shown here).



Left: Other experiments in page makeup have followed the approach of displaying headlines in readable characters,, while the body text is shown as lines of the appropriate lengths. Tektronix, a manufacturer of "storage display tubes, demonstrates how the capabilities of their terminals could be used for page makeup functions.

Below: Several manufacturers have succeeded in displaying actual type faces on their "soft copy" typesetters. This one—Compugraphic's PreView—utilizes the power of the Video Setter Universal to display a "soft proof" copy on the 19" viewing screen, showing actual type faces, sizes and position exactly as they will appear in typeset form.



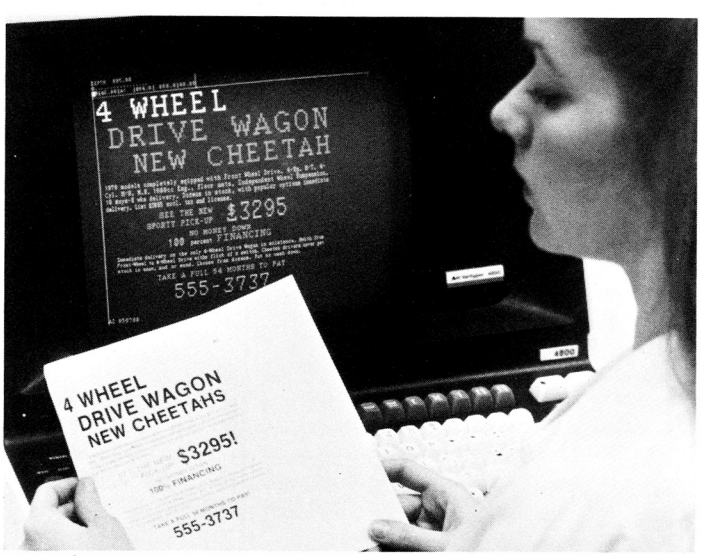
Above: Raytheon has also experimented with "filling the news hole." They are aiming for a closer representation of the final printed output by displaying actual characters on the face of the tube. In this trial effort, which has involved interfacing Raytheon's "CAM" terminal to a large-scale IBM computer, it was also possible to display a representation of a halftone as well as textual matter.



It is interesting that since then the major developments which have occurred have been precisely in the areas we predicted. And within the near future, very responsive interactive page makeup solutions, both for book work and for newspapers, will no doubt come to fruition. The technology is already with us. Some refinement of the definition of the tasks is still indicated. And as for magazines, that technique presently exists for those magazines which are written to fit assigned space, and copy-fitting takes place to help achieve this. *Time, Newsweek, U.S. News & World Report* and others regularly compose full pages, with all elements, including halftones, in place.

Review Questions:

1. Describe briefly the history of computer-assisted page makeup as it has evolved over the last two decades.
2. Discuss the ramifications of dealing with each of the following elements in an automated pagination program:
   - multi-column material with the same primary leading throughout
   - multi-column material set with varying values of primary leading
   - footnotes
   - tables
   - illustrations
   - headings and sub-heads
   - "no-break" items
   - poetry
3. What is vertical justification? How does it compare in scope and complexity with horizontal justification?
4. If you were responsible for designing a vertical justification program, what features would you incorporate in the package?
5. What is the difference between "automatic" page makeup and "automated" page makeup, as we use the terms?
6. What is the difference between an interactive display device and a "soft copy" display?
7. Find out what you can about the display ad terminals and interactive page makeup terminal systems currently on the market. Describe each system briefly.



In early 1979, AM International introduced the AM 4800, which is the Harris 2220, adapted, by agreement with the Harris Corporation, to output to a Comp/Edit or a Comp/Set 3500- or 4500-series direct-input typesetter. Its capabilities are precisely those of the 2220; the hardware is manufactured by Harris. But AM was given the right to market it in conjunction with the Comp/Set and Comp/Edit for either newspaper or commercial applications.

22

The Generation Of Halftones

IF AND WHEN the problem of page layout and interactive makeup is resolved, and complete, error-free pages can be composed effectively, attention shifts with even greater urgency to the incorporation of art work and illustrative material so that it can be "composed" along with the text, and stripping and all subsequent camera work can be avoided altogether.

Interest in this subject has already become stronger as more and more typesetting systems are in fact able to cope successfully with the arrangement of text within the page, thus avoiding the need for hands-on makeup. But if "holes" or "windows" must be left for the insertion of illustrations, some handwork remains, along with the possibility of error, and the failure to achieve the maximum amount of "editorial control" over the final product.

Yet another aspect of such editorial control would be the convenience of giving editorial people not only the option of placing the correct picture in the correct location, but also of selecting and sizing pictures—and the other illustrative matter—*interactively*—as judgment is applied to the consideration of the total impact upon the reader of a happy arrangement of pictures and text.

Thus, the intent of the publisher lies not merely in his desire to "set" the illustrations, but also to store them, retrieve them on demand, and manipulate them. It may also embrace the ability to compose or re-compose the entire page at remote locations.

Even in the absence of such interactive manipulation there are benefits to be derived. Composition of telephone yellow pages, for example, requires the extensive use of "logotypes." And who is to say where a logotype ends and an illustration begins? Whenever the same picture or design needs to be re-created frequently—for subsequent issues or alternate versions—a method which composes the desired art on demand has high utility.

A similar example, involving the use of halftones rather than line art, is the composition of home owners' buying guides (multiple listing directories). In these

two instances (yellow pages and multiple listing home guides) an interactive relationship between "editor"—or layout man—and his equipment is hardly necessary. But for the composition of a newspaper, magazine, or even a book page—using the term "composition" in its broadest and best sense—it would surely offer a powerful and satisfying capability.

Three aspects of the problem

We shall be considering three aspects of the problem of "generating" halftones and line art. They are:

- Getting the picture "into the system."
- Storing and manipulating the picture.
- Outputting the picture.

Obviously there are overlapping and interrelated aspects to these three topics, and so our treatment will involve some digression and some jumping back and forth. We have also found it most convenient to postpone the second aspect of the problem—"storing and manipulating the picture"—until we have given some consideration to the third topic—"outputting the picture."

Getting the picture into the system

In most instances, in order to "input" the picture, "scanning" is required, and you have a choice of scanning techniques, just as you do with OCR.

1. ***You could use a vidicon tube or a TV-type camera***. The image picked up by the camera lens is brought into focus on the face of the tube, where a raster scan by a directed electron beam sweeps the tube in a predetermined pattern, and "reads" what it "sees," reporting signals which carry varying degrees of light —and hence electronic—intensity. These signals—initially, at least—are of an analog nature. In other words, the electron beam within the vidicon camera senses and conveys not merely black or white observations, but a wide range of tonal values, If these values are retained in the analog form in which they are received, perhaps stored, and subsequently—or immediately—output, then the entire system will be of an analog nature.

This was, in fact, the principle behind the design of the Linotron 1010—a "Lexical—*Graphical*" CRT typesetter produced under a contract issued in March, 1964 by the U.S. Govenment Printing Office. In addition to the two 1010's to be designed and built by Mergenthaler in conjunction with the Columbia Broadcasting System Laboratories for the GPO, the companies undertook to deliver two units to the Wright-Patterson Air Force Base with the capability of outputting *line drawings* (and even halftones) from *video* tape, using the same CRT as the typesetter for the display and exposure of images from data recorded and stored on a video tape deck as analog information. This feature was not, in fact, operative until 1971. It was still being used in 1979, and while some halftones have been created for military publications, line drawings are composed commonly.

The Lexical Graphical Composer Printer

(Linotron 1010)

and the "Graphics Channel" related equipment

While this is "ancient history," very little has in fact been published about the graphics capabilities of the Linotron 1010. Consequently, on this page, and the next two, are reproduced a diagram of the "system" and test samples of work produced thereon. The halftone which is reproduced was pre-screened. Both sets of samples were generated by the videotape method.

The Graphics copier scans master artwork with a fine beam spot of light, converting each small picture element into a video signal which is recorded by the video tape recorder. This video signal record, containing a graphics identification number, sync signals, and other data which the copier also records when the artwork is scanned, serves as the recording which is played back in the Graphics Recorder. Since videotape recordings consist of magnetically stored data, having indefinite shelf life, the recorded tape can be played back many times. However, long term erosion of the tape surface by rubbing during tape motion precludes indefinite usage, as first random noise and then dropout occurs.

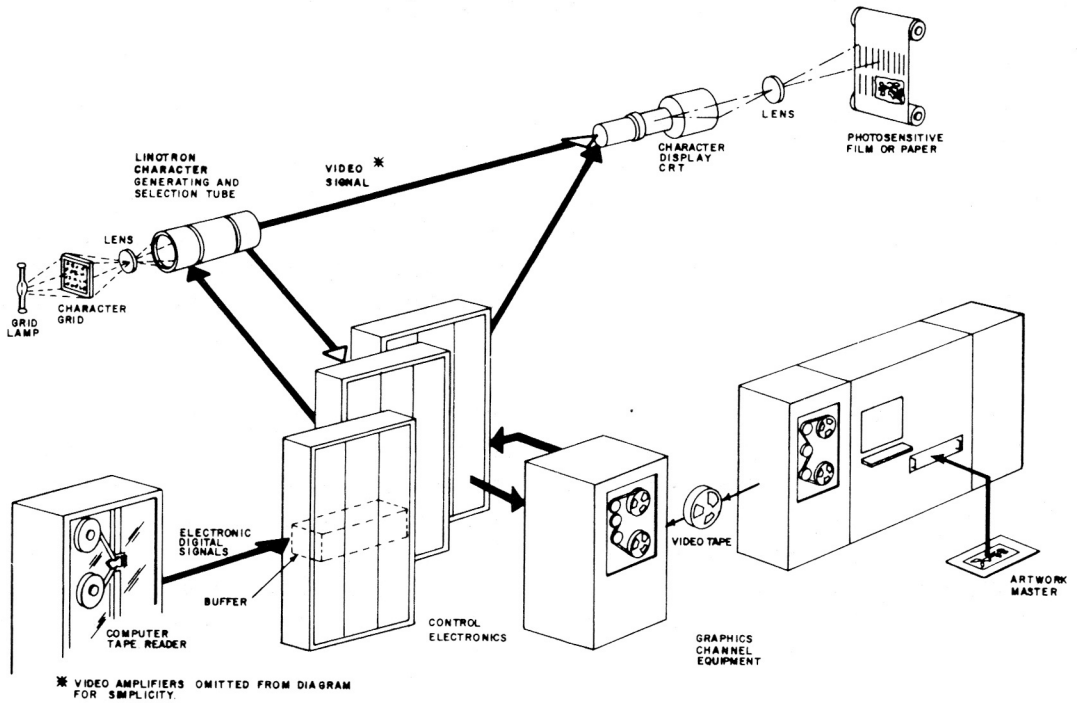
Up to about 250 to 270 Graphics Copier recordings can be made on a one-hour videotape. Any or all of them can be used in a single run if, when two or more are desired, the computer tape requests them in the sequence in which they are placed on the videotape.

In the scanning process, the light spot is projected on the master through a lens from the screen of a special-purpose cathode ray tube. This tube has a fine beam of constant intensity and acts as a high-resolution flying spot scanner. The moving spot of light from the beam continuously illuminates small, discrete portions—or elements—of the picture, and the reflected light serves to excite an array of photomultiplier tubes (PMT's) whose current outputs are summed in an adding network.

The large unit of the Graphics Copier group of equipment contains a light-tight drawer where master art work lies (one at a time) during the scanning conversion to video signals. The master image may consist of monochrome original art or a reproducible copy of original art (preferably a photograph), and it can be effectively reduced or enlarged up to 10% as determined by the use of copier controls. The copier controls also provide for a wide range of screening-dot densities to be provided by the Graphics Recorder when the videotape is being played back. This preselection is recorded with the video data on the videotape in the VTR. One of these densities —or "screening rates"—must be chosen for any halftone composition to be made on the output medium in the LGCP. (A LINE ART preselection equivalent to "no screen" is necessary when line art is placed in the drawer.)

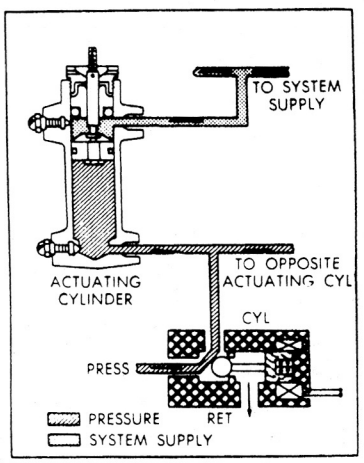
—Excerpted from Mergenthaler manual 4/30/71

The Lexical Graphical Printer (Linotron 1010)

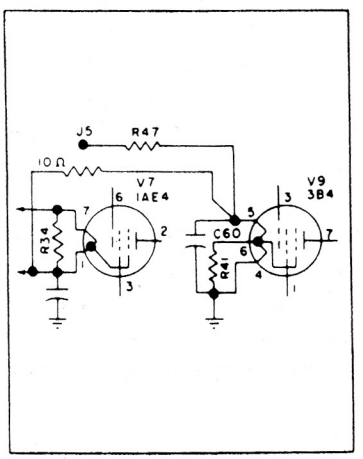


Simplified diagram of the Lexical Graphical Composer Printer.

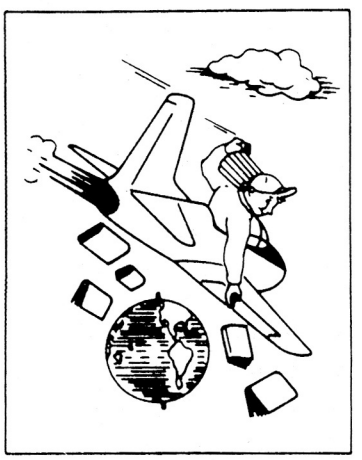
Line Drawings Output on the 1010



Schematics



Diagrams



Cartoons

This line art was "typeset" by the Linotron 1010 during its testing cycle in 1971. The images were written onto the full-face CRT from videotape which serves as the recording medium. We believe the two Linotron 1010's delivered to the Air Force Base are still in use today and do in fact produce graphics—especially line art—primarily for manuals and other documentation.



This halftone was generated on the Linotron 1010, subsequently shipped to the Wright Patterson Air Force Base, during a final testing cycle at Mergenthaler. We were given a film positive, which our printer contacted (with some loss of detail) to obtain the necessary negative. We do not know precisely what screen was used.

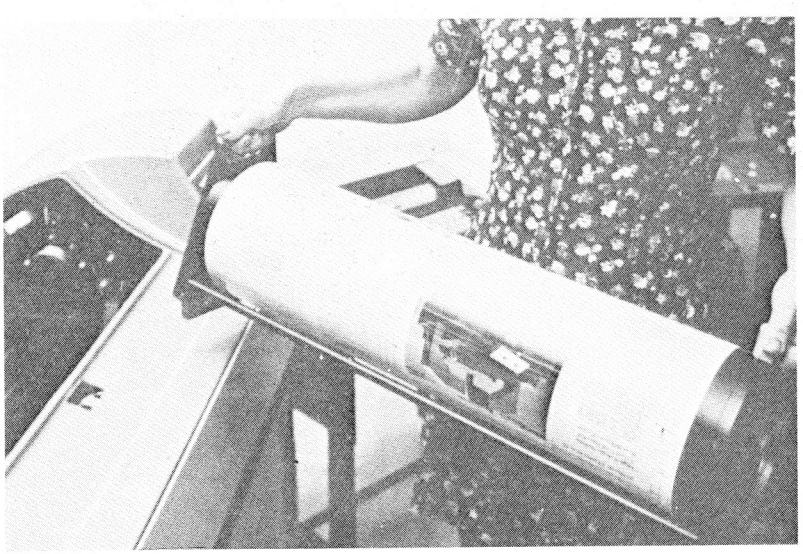
But if these analog values—such as might be written to video tape—were subjected to analysis in a "a-to-d" (analog-to-digital) converter, they could be transformed into digital information, either as black and white "observations"—in which case they could be recorded as bits (0 or 1), or as shades of gray. In the latter instance they would be written as "bytes" with perhaps up to 256 different tonal variations, from "all white" to "all black."

In either case, the values read (as by a vidicon tube) are subjected to a "threshold" test. If the intensity of the signal fails to reach a certain level (in a two-tone system) it is recorded as "white"—or zero. If it reaches or exceeds that level it is recorded as "black"—or "1." However, the levels of intensity can be translated into a graduated scale of steps, each determined by its own threshold value—in a linear relationship or according to some preferred curve. Gray scale levels, as from 0 to 3, 0 to 15, or even 0 to 255—can thus be registered. Moreover, these threshold values can be altered from time to time in order to enhance or reduce the contrast scale (the slope of the line), or the midtone values (the shape of the curve) or both.

2. ***You could use a flying spot scanner or some other method of focusing a light beam on a specified location***—or, more precisely, upon a series of locations which can be "tracked"—usually in a raster pattern. Then this spot of light (probably as reflected and "sensed") can be "read" by one or more photo diodes, and its intensity—in like manner—can be recorded as analog information, or translated into digital form—either as black and white or as shades of gray.

Getting light on the subject.—There are, of course, various methods of directing a spot of light onto the subject (picture). You could use a flat-bed scanner, where a beam tracks across the image a "line" at a time—and these "lines" can be very fine indeed, such as 1/1000 of an inch apart. Or a drum scanner could be used, where a reflected beam is read from a fixed location as the drum revolves, and then either the drum moves to record another series of observations, or the beam moves. The beam can be a coherent light source, such as a laser, or it could be a relatively tiny spot of light played across the picture by means of a series of holes in an otherwise opaque revolving disk, or a revolving mirror which sprays spots of coherent light across a surface. The "sensing" of the observations is usually performed by one or more photo diodes.

One of the devices frequently used for digitizing graphics (as well as for writing out scanned patterns) is the Optronics scanner/plotter.



The scanning resolution

However the process takes place, and whatever means may be employed to record the data—whether in analog or digital fashion—the "resolution" of the information is a function of the number of discrete observations made—or

"samples" taken. On a vidicon tube the maximum resolution will be determined by the raster sweep of the scanning beam and the granularity of the phosphor which is excited by experiencing the light rays of the image and the "play" of the electronic beam. For a "coarser" resolution, some of the observations, such as each alternate one, could be "thrown away."

When a flying spot scanner, or a flat bed or drum scanner, is used, maximum resolution is determined by the mechanics of the movement of beam, bed or drum, and the size of the spot produced by the light beam. One significant difference between existing systems or techniques has to do with the number of observations taken over a given area. It is possible to take such observations in both horizontal and vertical directions every 1/1000 of an inch, but some systems, seeking to scan a continuous tone image and to generate a halftone, may make their observations only according to the screening resolution desired. (More about this later.) But if we do assume the 1/1000 inch scanning pattern, then, presumably, the spot to be "read" and recorded will also not be larger than 1/1000 of an inch (a "mil") and, indeed, may be somewhat smaller. But we shall assume, for the moment, that on the input side there will be 1,000×1,000 (which equals one million) observations for each square inch of surface scanned.

The simplest way to deal with this vast quantity of information would be to pass it on directly to the output device in "real time" as the scanning process takes place. If the output device, or "writer," offers the same resolution as the input device, or "reader," then each spot reported—with whatever its intensity—could be written out, as by a laser beam, and recorded on a sheet of photographic film or paper or onto a treated printing plate. But, as we shall see, it would also be possible to read at one resolution, and to "write" at another. (This might be the case if we wanted to enlarge or reduce the output relative to the input, for example.) The output resolution would thus differ from the input resolution, being finer or coarser, as the case may be.

For convenience in our subsequent discussions, the minute, spot-sized observations, consisting, perhaps, of 1,000×1,000 dots, each one mil in diameter, may be called "pels" or "pixels" for "picture elements." Some people prefer one term and some another. Some people may use the same term for what is written out, even though what is read in is not necessarily the same thing as that which is written out.

Facsimile

The reader will have noticed that the process we have described—both of reading and of writing—is one of *facsimile*—that is, of seeking to create a facsimile image of the picture or other "copy" being scanned. But facsimile reproduction usually implies one other factor—namely the transmission of the information from one location to another. The thing *read* is in your office. The facsimile version is recorded in someone else's office. Facsimile reproduction usually occurs, however, in real time. The image transmitted is recorded (written) from data which is not stored but merely passed on as it is created by the reading

process. The output device of a fax unit may consist of hammers which deposit carbon from a ribbon for each "spot," or of an electrostatic transfer of ink. But it could also be a laser-borne spot of light conveyed to a photosensitive substance.

Continuous tone vs. line art

It is possible, of course, to scan continuous tone photographs, or line drawings, or type—all in the same fashion—and to reproduce them in the same manner. But a continuous tone image, when scanned, would be recorded as a continuous tone image. It might appear "screened" if the writing unit's resolution and spot size are too gross, but to get good quality you would want as much resolution as possible, and as fine a spot size as possible. Thus you could reproduce such output on a printing press no more successfully than you could reproduce the original photograph without "screening."

In order successfully to transfer ink from press to paper, a screen process must be used. (There are a few exceptions, where the grain is imparted in some other way, such as *Collotype.*)

The screening process transforms the continuous tone image into a series of "dots" or "cells." The nature and size of the cell will be determined by the kind of screen used. Halftones are created with a screen which may range from perhaps 55 to 133 or more lines per inch. A 55-line screen is coarse, and 85- to 110-line screens may be considered "medium," while screens with a ruling of 133 or more lines may be called "fine"—especially for letterpress or offset printing. Gravure screens may offer rulings of 120, 150, 166, 175, or even 200 and 300 lines per inch. When we talk of a 110-line screen (for example) we mean a screen with that many lines in what we may loosely term as both vertical and horizontal directions. Actually, black and white pictures are reproduced almost universally at a screen angle of 45°. This is done, according to Brian Chapman of Printing Developments, Inc., "because of the physiology of human vision. Any other angle is more easily detected by the eye as a halftone."[1]

Such a screen is, in effect, interposed between camera and film—directly in front of the film and in contact with it, or nearly so, so that the solids are broken up. (This is an oversimplification, because there are crossline or glass screens, contact screens, and prescreened films, for example.)

It is not necessary to screen type or line art (except for the gravure process). unless there are very heavy solids, in which case a screen or screen effect is often used, such as a Benday or tint, which might vary from perhaps five or ten percent to perhaps 90 or 95 percent—to convey the effect of a light gray shading or a solid. As a general rule, whenever you have large areas of black, some screening is desirable, and in a halftone, for example, your lightest "dot" might be two or three percent "black" and your blackest dot could be five percent "white."[2]

Thus, if we are scanning a photograph for reproduction by means of "typesetting," or directly onto film for platemaking, or directly onto a printing plate, we

1. In a letter to this writer.

2. Dr. William F. Schreiber, of M.I.T., states in an article entitled, "Production of Images on Printing Presses":

". . . Obviously, if it were possible, we would choose a very fine screen in all cases since that would give imagery visually most like the original. Since we depend on the eye to average the array of dots to produce a similar effect to viewing the original, the finer the screen, the more closely the image could be examined without the dot structure becoming obtrusive. Of course, the image cannot be whiter than the paper or blacker than the ink. But within this range, the dynamic range is limited by the smallest black and white dots which can be printed reliably. This in turn depends on the ink, press, paper and skill of the operator. Using the best coated paper with all other factors optimized, printing press results can be about as good as photographs on glossy paper—100 to 1 (2 log units) or slightly better. Newspapers, however, barely achieve 10 to 1 (1 log unit). In letterpress printing, an additional problem occurs with isolated highlight dots, which are bound to occur if dots are allowed to drop out in any light area. Such isolated dots tend to accumulate globules of ink and print very large. For this reason, letterpress printers usually try to keep a minimum dot everywhere. Lithographic printers, both offset and direct, may or may not keep dots everywhere, but in both cases, there is a smallest reliable dot. To achieve a given dynamic range with a given minimum dot requires a minimum dot spacing."

must contrive some method of screening the picture electronically, unless we want to scan an image which has already been screened.

The latter is possible, of course, but not usually desirable, because it eliminates the possibility of subsequent sizing (you would need to change the screening resolution). It might also introduce a *moiré* effect by interposing one dot pattern upon another. So it is clearly better, it seems to the writer, to scan an unscreened photograph, and the best place to generate an electronic screen is to do so *after sizing decisions have been taken.* This is a point to remember, since interactive manipulation of halftones would appear to be handled best while they are still, in effect, electronic "continuous tone" images.

We have gone into some of these points—all too well understood by most printers—primarily to make clear that there is a confusion in terminology. The "dot" which is scanned or written out is sometimes not a halftone dot. It is a "pel" or "pixel" on the input side—which may be the same as a halftone dot, or may not be. It is often a halftone dot on the output side, but it may be made up of pels or pixels. So far as output is concerned, the halftone dot isn't really a dot at all. This phrase is a misnomer, but probably too well entrenched to get rid of. It might be better termed a halftone "cell."

The output device

Before we discuss problems associated with the storing and manipulation of the scanned data, and its transmission to the output unit, let us consider the output device itself. It will be either a CRT typesetter, a "laser typesetter," or some form of "writer"—probably consisting of a directed, coherent laser light source which can be turned on and off.

If the device is a CRT typesetter, it will probably not "write" its output in a "raster scan" mode, although this is, of course, a possibility. Its beam—unlike the usual TV tube—can be deflected at will from external signals. It is likely that the beam will stroke type line-by-line, vertically—that is, creating vertical lines one to two mils wide, at a resolution of 500 to 1400 lines per inch. The software or firmware to drive such typesetters tells the beam where to turn itself on or off while sweeping across the face (or aperture) of the tube. When the beam is turned on, it is usually "tuned" so that it produces a spot or dot of a certain intensity, or no dot at all. In other words, it is not its function to produce shades of gray. It is possible, of course, to vary the intensity or the energy of the beam, causing the spot size to be larger or smaller, and more or less bright, but it would be difficult to do this on a dot-for-dot basis, though not so difficult on a "character-by-character" basis, where characters consist of a series of contiguous strokes. In any event, the dots which the CRT writes out are more like "pels" or "pixels" than they are halftone dots. A halftone dot for a 100-line screen would have to be up to 10 mils in diameter. The usual CRT spot size is one or two mils. Hence, to get a halftone dot or cell you would have to create that "image" out of a series of CRT dots or strokes, using as many as perhaps 100 pels or pixels, or perhaps 10 vertical strokes, where the light beam has been turned on and off several times per stroke in order to create the desired photographic halftone cell pattern.

For line art, on the other hand, as for the generation of type, no screening process is generally required, although a screening capability might be useful to provide Benday effects and to make large solid areas easier to reproduce. It could be helpful, too, if the CRT had the means to create "reverses"—to leave the character shapes white and to surround them with black. (The Seaco CRT—which did not reach production beyond the prototype stage—and the MGD Metro-set typesetters could do this, as can the VideoSetter.)

Therefore, in addition to the setting of type, we have identified certain other functions which *might be* performed by a CRT typesetter:

- Reproducing line art, presumably on a dot-for-dot basis.
- Creating Benday effects and/or reverses.
- Laying down "cells" of the desired resolution for halftone dot or cell generation.

It might also be useful for the CRT typesetter to have the ability to create continuous tone facsimiles, for subsequent screening off-line, in case anyone wanted to use the same device as a "remote printer" and to screen the images, or some of them, to meet the requirements of press and plate at the distant site, when such requirements might vary from site to site.

A "laser typesetter" as an output device

Presumably, a "laser typesetter" such as that briefly offered by Dymo—and more recently by Monotype—could perform the same functions as a CRT typesetter, including halftone generation.

The Dymo DLC-1000 typesetter or the Monotype Lasercomp could offer some advantages for graphics generation. In the case of the Dymo machine, the laser beam was deflected vertically to create a stroking pattern, and then these strokes were "sprayed" by means of a rotating mirror. The beam could be turned on and off to provide line segments necessary to draw characters. Consequently, the effect was very much like that of stroking characters in the "window" of the CRT tube.

With the Monotype Lasercomp, there is no vertical laserbeam deflection, although the laser is deflected horizontally across the entire width of a 58-pica line by the movement of a multi-faceted mirror. Each dot of the character image is laid down in a raster scan pattern. As of this writing, the Dymo product has ceased to exist. Neither Dymo nor Monotype has tackled the problem of halftone generation.

The CRT device would appear to have the advantage of greater flexibility in the manipulation of the writing beam, in the sizing and spacing of dot patterns, and perhaps in varying the intensity of the dot image. Also, the vertical stroking pattern of the Dymo laser typesetter would have made it necessary to create any halftone larger than one inch in height as a series of slices. This limitation would

apply to any CRT typesetter which composes only in an aperture mode, though it would not be applicable to a "full-face" CRT such as the Information International VideoComp 570 with full-face option.

But it is possible, of course—at least in theory—to vary the stroking pattern of either a CRT or the Dymo laser typesetter so that the device would assume a raster scan mode for halftone generation, if that mode is, indeed, desirable.

Whether the desired screening effects, reverses, and other kinds of manipulation, including halftone generation, would be performed by the "intelligence" of the typesetter, or by some other intervening or superior intelligence—*e.g.*, in the composition computer—would be a matter of system design. For halftone generation the VideoComp 570 is provided with intelligence to form its own photographic characters.

The use of a laser writer for halftones

It would appear that a laser writer could offer many or most of the capabilities one might desire in an output typesetter and character generator. Such a device would presumably not "stroke" characters, but would generate them as dot patterns, laying them down in a raster sequence, as does the Monotype Lasercomp. (Not that the Lasercomp has yet tackled anything other than type!) The character repertoire for halftones would presumably be stored as permanent patterns or "characters," whereas line art would be generated at the pixel level. Logotypes can, of course, be digitized as if they were type characters, but this concept may not work when it comes to larger illustrations. At present, logotypes are indeed treated as type characters, being stored and digitized in the same manner. For example, the users of the Hell Digiset have available to them the Digigraph 40A20 which can handle line art up to the size of an A4 page. It is also able to process pre-screened halftones.

The point here, however, is that *all* material, including type, regardless of how stored, could be *formatted* in memory so that it could be composed by means of a raster technique.

Unlike the CRT typesetter, it will be noted that a laser output writer could be used either for creating images on photographic paper or film, on the one hand, or photographic plates, on the other. Either a flat-bed scanner/writer, or a drum device, could be so used.

There are, in fact, a good many related bits of technology presently available. For example, there are laser platemakers, which scan a pasted-up page, and which, either remotely or on-line, write the scanned image onto a photosensitized plate ready for press.[1] These plates can contain screened halftones as well as line art and type. This technique is particularly attractive for newspaper-quality generation of printing plates at remote locations, from a pasted-up master page created at the editorial site.

1. Eocom and Dest Data are cases in point.

There are also such devices as the Hell Helioklischograph, which scans color separations and engraves these images on gravure cylinders, as well as the Crosfield MagnaScan, a similar machine by Linotype-Paul, and the PDI color scanner. The Pagitron, by Optronics International, is perhaps the device which most closely resembles what it is that we have suggested as a general-purpose output device, although the Pagitron also offers interactive makeup capabilities which go beyond this.

It seems to us pretty clear that the scanning and writing technology which presently exists is at least adequate for the purpose of typesetting *and* image (*e.g.*, halftone) generation. Remaining problems have to do with configuring an appropriate system, and the storage, transmission and manipulation of information—whether by hardware, firmware or software. This generalization is not entirely valid, however, and there may well be an element of wishful thinking, since to manipulate graphics *interactively*—including text—implies that you either degenerate the quality of the image as you deviate from its "true" size—or else that you must work with an abstraction of that image until you have made up your mind what sizes and shapes should obtain, and then, somehow, manage to create the desired raster pattern. But this speculation is too esoteric to pursue at this point.

For the balance of the present chapter we shall assume that digital techniques will be used—although an analog system would seem to be preferred if the reading and writing functions were to be performed simultaneously at the same location.[1]

For data storage and transmission, digital techniques seem preferable because:

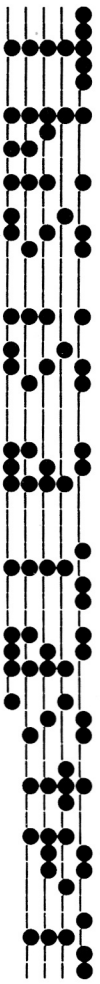
- of the possibility of data compression; and
- type characters can be communicated as bytes rather than as "pels"; and
- photographs can also be stored and transmitted as "characters" or cells rather than as "pels." In other words, the amount of information in a 100-line screen halftone one-inch square can be described as 10,000 "bytes" —80,000 or fewer "bits"—rather than as 1,000,000 pels (or the equivalent in analog information).

Obviously, when it comes to type, the information can be handled most effecively as *characters* rather than as scanned art work. The output writer or typesetter must therefore take on the task of translating type characters into CRT strokes, or laser-writer raster dot patterns. RAM memory can be used to format lines of type so that an entire line at a time can be composed in a serial fashion, just as characters are presently created by an electrostatic printer, only to a finer resolution.

Because pels are too small and too close together to reproduce without "screening," the task is to convert a small square or cell of picture elements into a single "character." If we wish a 100-line screen, then each character might consist of a matrix of 100 pels, each 1/1000 inch in diameter. A 150-line screen would

1. Linotype-Paul has, in fact, offered an on-line, real-time method of scanning 35-mm. slides (or portions thereof) for the generation of logotypes and line art generally. When the illustration is desired, a carousel containing the slide in question is rotated to the proper location in a scanning device, the slide is scanned, and the image, appropriately sized, is generated on the output CRT. It must be remembered that this device was intended to work in conjunction with the Linotron 303 —an analog imaging CRT typesetter.

be comprised of a 7×7 or 8×8 matrix of pels. We shall consider these matrices "photographic characters"—and we shall now consider their structure.



Creating photographic characters

If we are considering the possibility of transmitting complete pages—including photographs—for remote setting, a case could be made for transmitting characters as characters and photographs as pels. The writing device would lay down the characters and then return to the windows where photographs are to be positioned, read the "pels," and convert them into photographic characters for output, with the screen resolution and halftone dot structure being determined as a function of the platemaking and printing requirements for a plant in a particular locality. This would permit a publisher to transmit identical pages for reproduction in one location by letterpress, and in another location by offset, or direct-to-plate in a third.

More likely, however, the picture portion of the page will be transmitted and/or stored as "photographic characters" since less information storage or transmission is involved.) But in either case, the task is to create such photographic characters from pels or strokes of some nature.

What should a photographic character look like? There seem to be several schools of thought in this respect. One says that it should look exactly like—or resemble as closely as possible—a halftone dot created by conventional photographic techniques. The other school seems to be saying that there is nothing sacred about the halftone dot structure that just happened to emerge from the use of photographic techniques. Conceivably, one can generate an alternate halftone cell structure by computer program which yields dot patterns—and hence produces final (printed) output—superior to that which is the result of the labor and traditions of our conventional wisdom. John J. Tuohey, vice-president of manufacturing at *U.S. News & World Report*, wrote us concerning a draft of this chapter. "I like your discussion of the 'dot' *vs.* the pixel. Theoretically, at least, we ought to be able to get a better press product (*i.e.*, illusion) from a CRT output than from conventional screening methods. And the only way we will find that out is by trial and error. The film is just an intermediate step and comparing dots under a glass in this case is meaningless."

But what happens when a photograph is screened is not just that the solids are broken up. Three other things occur. First, you get the desired gray scale, since you are picking up (unless you wish to enhance them) the continuous tonal values of the original picture. If you reduce these gray levels to any finite number, even 256, presumably there is some sacrifice in rendition, although this may be debated.

Second, you get a certain *contour*. The dot shape or cell shape still presents *not just a tonal value*, but conveys a representation of the shape (in that minute area) of the subject. It may not be much of a shape, but it seems that *the coarser the screen, the more important the contour will be*. The finer the screen, the less significant the contour factor, since each cell will itself, by its shape contribute to

the contour. That is to say, the closer you set to the pixel level, the less concern you have about the *shape* of the photographic character represented by the cell itself, and the more you are concerned simply with the gray scale level.

Professor William Schreiber tells us that, in his judgment, you get "true" sharpness with respect to contour at about a 400-line screen, when using unmassaged pixel elements. Actually, it is much more complicated than that since gray levels also get involved. "Assuming that the pels from which the dots are constructed represent the smallest spot which can be printed by the printing press, it can readily be seen that in selecting the screen gauge or dots per inch, we are effectively trading off resolution *versus* the number of gray levels. If the pels are 722 per inch, the screen gauge is 65 dots per inch as measured diagonally between the center of adjacent black dots. A 6×6 brick (*i.e.*, cell) would give an 85-line screen with 73 gray levels, and a 5*m*10 brick, a 100-line screen with 50 gray levels."

Brian Chapman, of Printing Developments, Inc., a Time, Inc., subsidiary which has developed, among other products, an electronic color separation process, writes: "We have spent a great deal of research money (several millions of dollars) improving so-called conventional engravers' dots. (I like your term 'cell' much better.) In fact, our screened images are easily differentiated from conventionally-screened dots—even by a layman. We distort them, push them out of position, line them up on sharp contrast edges, and use many other techniques to improve the picture's fidelity. It is for this reason that we must use almost as much electronic information (analog and/or digital) to form our screened images as we do our continuous tone images. I believe this high-fidelity approach to halftoning is going to be painfully learned by some of the typesetting people, particularly in view of the deteriorating state of affairs with regard to paper and ink. There is always the possiblity that magazine pictures may continue to decline in quality in the name of expediency, but this remains to be seen. (In color separations PDI uses conventional screen angles simply because any deviation from this gives size increase in the so-called 'rosette'—actually *moire*—pattern.)"

Returning to our analysis, the third thing that happens in the screening process is some change in dot shape which goes beyond the representation of contour. Something happens when you form halftone dots by photographic means. This is perhaps explained by what is known as *diffraction theory*. As Victor Strauss explains it in his invaluable textbook,[1] "Broadly stated, each transparent area or opening of the crossline screen acts like a minute lens, and each opening forms a minute image of the camera lens aperture on the film surface. These images are not sharp, and as they also somewhat overlap, the result is a halftone dot pattern. According to the diffraction theory, each different screen ruling should be placed at a specific distance from the photogtraphic material in order to obtain the best possible halftone image.

The lens itself contributes to the shape of the dot. The use of an iris diaphragm produces a circular lens opening, and the amount of light which goes through the screen is of course a function of the size of the lens opening, and this

1. Victor Strauss, *The Printing Industry*, 1967. Published by Printing Industries of America, Inc., in association with R.R. Bowker Company, page 187.

helps to determine the size of the dot. The screen itself leaves a series of pinpoint dots in white areas, which may have to be eliminated by a "drop-out" technique.

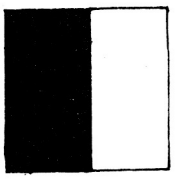
Whatever transpires during the conventional screening process, it appears that the results need to be understood. Still, in the last analysis, the proof of the pudding is in the eating. And so you will have to judge whether the photo-engravers' dot is best and should be emulated, or whether special electronic manipulation is required. One concept, which occupies a kind of middle ground, suggests that an analysis of the observations that would comprise one cell, could produce a character repertoire not only for each given gray scale level, but within one gray scale level, a series of shapes might be presented to improve contour.

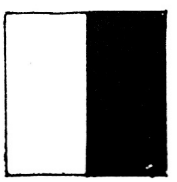
The gray scale

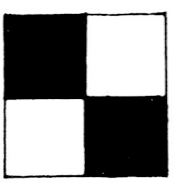
Aside from the matter of resolution—or screen—there is the problem of gray scale. Conventional photographic processes do not in fact impose a "level" of gray—although there is talk of a 5% dot. The imposition of gray scale levels is a function of the digitizing of halftones. Such levels are not required by analog processing. To achieve proper digital levels you will need to know, of course, how much graduation and for what purpose? If you decide that you require 256 levels you will need to describe each level with an eight-bit byte. If you need only 64, a six-bit representation will suffice. We have seen some reasonably acceptable halftones with only four levels of gray! This would mean that only two bits would be required to record and transmit each photographic character. We are quite certain that more than this is, in fact, necessary for an adequate representation of the quality of a continuous tone image as a halftone.

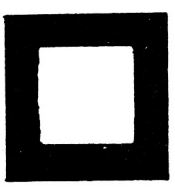
Is it then possible to shape the dot not only to suggest contour, but to represent the appropriate gray level values? In principle, this could be done. Suppose, for example, we wish to describe a 50% value. Would it be better to offer a character repertoire which contains not just one character for a 50% combination of black and white, but rather a series of such characters, one of which may be more appropriate than another under the circumstances?

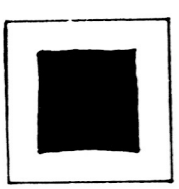
If you decide that you require more than one character for each tonal value, how many do you need, and when would you use a particular one? (If you had 256 gray levels and eight basic characters for each level, you would need to transmit 11 or 12 bit "bytes" or use "font shift" encoding. The former would probably be faster, and still better than transmitting "pels."



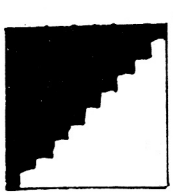


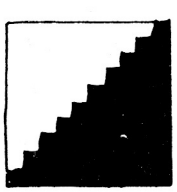


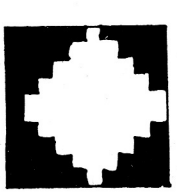


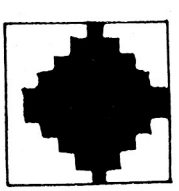


These are some possible "photographic characters" from a photographic character repertoire, all representing "50 percent dots," but available to indicate desired contours. A 10 x 10 matrix of pixels is assumed here.









It appears that the only way for the program to "know" what the dot shape should be is to analyze the pixel structure of a group of related pixels to try to simulate the appearance of a halftone dot as closely as possible. However, one school of thought suggests that the halftone dot structure should differ from that which would be constructed from the original continuous tone (or from pixel samples of the original) because of the tendency in conventional screening for dark areas to move toward dark areas and light areas toward light areas—thus enhancing contrasts and changing shapes somewhat. This effect does in fact occur in conventional screening, as we have already mentioned, partly because of optical properties of lenses and light rays, and partly because of film chemistry.

It is, of course, not beyond the capability of "software" to modify dot patterns and to generate odd-shaped photographic characters both with respect to their shape and their tonal values—either as a result of "proximity analysis" or through interactive, manual intervention.

Modifying contrast and tonal values

Software can also enhance the overall contrast of the output (in relation to the input) by changing the slope of the gray scale curve. A reading of the original might indicate a contrast ratio of 10:1 between the darkest and the lightest areas. This could be increased to 20:1 or even 40:1 by altering threshold conversion values, as well as by interpolating new values.

Changes in midtones and highlights can also be made selectively, according to the requirements of individual pictures, by setting dials to alter the scales for reading or writing, or for the purpose of defining the structure and weight of the photographic characters themselves. A linear relationship between such values is not necessarily implied, and threshold levels can be fashioned along some sort of curve.

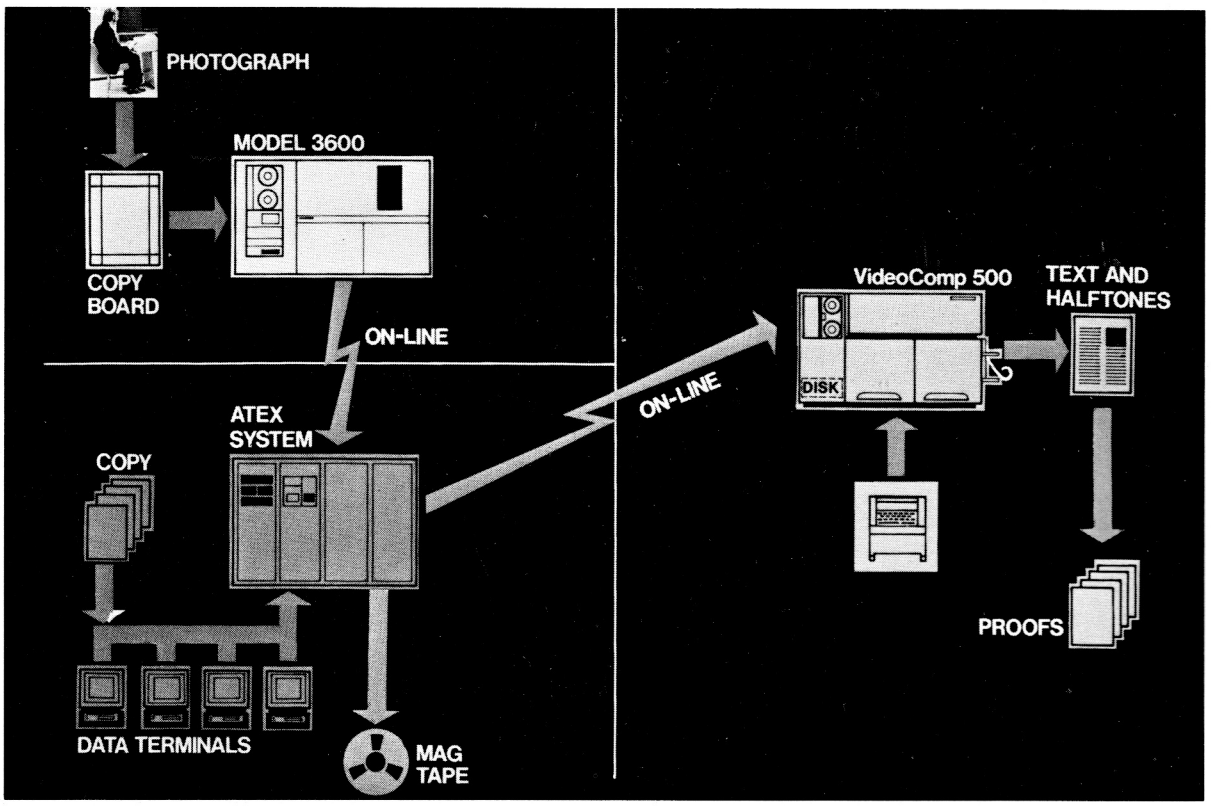
What is presently available?

In terms of the full page composition of halftones, along with text, the most interesting development to date is that involving *U.S. News & World Report* and its use of the Information International Model 3600 "Continuous Tone Scanner" in conjunction with the VideoComp 500 or 570. Since 1974, *U.S. News* had been producing full page composition on an Atex system, and relaying the data to Chicago for resetting (but not re-composition). Donnelley mounted a tape on its VideoComp, received by direct transmission, and reset the page which has already been composed at *U.S. News*. But Donnelley also had to incorporate the illustrative material.

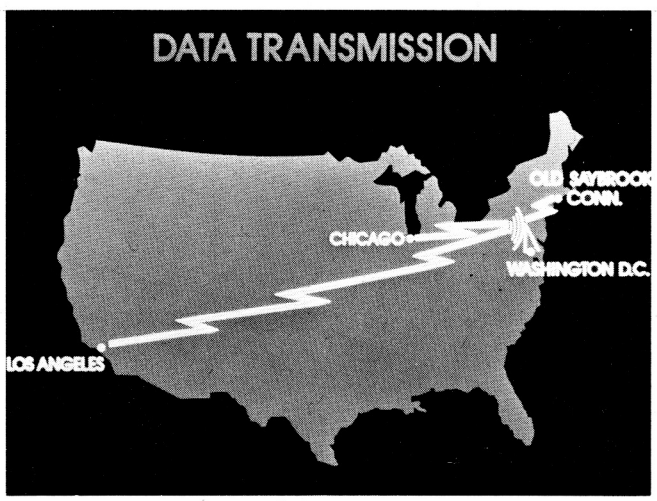
Page negatives were thus produced. These were flown to Los Angeles and to Old Saybrook, Conn., for the east and west coast press runs. Final pages were flown from Washington to Old Saybrook.

In 1977, *U.S. News* began to transmit complete pages, *with halftones,* to all three locations, *U.S. News* has its own VideoComp in Washington for the trial

setting of pages, and for other purposes. To generate halftones in position, it uses the III 3600 scanner. Art and photographs are placed on the scanning bed, where a flying spot scanner generated by a CRT tube moves across the image to be scanned. Reflected light is measured by two photo diodes. Their observations are



In the U.S. News system configuration, input from Information International's Continuous Tone Scanner is stored on disc and hence is relayed to the VideoComp. The satellite system network, which transmits fully-composed pages, halftones included, to Chicago, Los Angeles and Old Saybrook, Conn., is diagrammed below. *Right:* A picture to be scanned is about to be inserted in the 3600 Scanner.





averaged and recorded as a 256 gray-level scale. Scanning takes place for purposes of a 110-line screen, so that the area scanned (assuming no photographic reduction) is an area 1/110 of an inch square. Thus, in this case, the observations are not done at the pixel level, unless you consider a pixel and a halftone cell to be the same thing. III can go at least to 133-line screening, but the printing process (Nyloprint letterpress plates) prohibits the use of so fine a resolution.

The scanned information is written directly to disc (it could be stored on tape), and when pages are composed this information is incorporated into the data being sent to the VideoComp. The VideoComp sets text in the conventional manner, but when it receives the stream of data for the halftones—as one eight-bit byte for each halftone cell, the intelligence of the VideoComp decides what photographic character to typeset. The VideoComp has available to it a repertoire of approximately 300 characters, representing different sizes (for different screening effects) and different shades of gray. The present system provides 64 gray levels along a linear curve, but in fact something more than 64 could be generated. At a 65-line screen the gray scale would require a repertoire of 100 photographic characters or less, we are told, but at 133, it could be as high as 220. What this means is that for a 65-line screen, the program selects one set of characters. For 110 lines, it would select another, out of a total of 300 photographic characters the VideoComp will have available to it. It is noted that each VideoComp can be adjusted to provide the characteristics desired for a particular printing operation, it apparently isn't as easy as setting a dial, but the characters selected—and hence the amount of contrast, midtone values and the like—can be determined for each location depending upon the nature of the printing process and the kind of plate material used.

This is not an interactive process. If there is any cropping or reduction to be done (the scanner does not enlarge), it must be performed during the scanner input operation. It is true, of course, that the VideoComp has the capability of electronic sizing of character images, but this would obviously affect the screening and the quality of the output.

Final pages are then sent from Washington D.C. to Chicago and Los Angeles by Westar satellite, and to Old Saybrook by microwave.[1] Transmission is at 9600 baud, and four pages, consisting of 10% halftones—and the balance, of course, in type—can be transmitted in six minutes and 43 seconds! (Somewhat less time than it would take for the courier to get off the elevator.)

1. *U.S. News* planned, by the fall of 1979, to move away from letterpress and the Nyloprint plates, to increase the resolution of its halftones to 133 line screen and to change some of its suppliers, still utilizing the same typesetting technology.

Other techniques

While Information International, the makers of the VideoComp 570, introduced the 3600 Continuous Tone Scanner in 1977, there has been a scanning capability for the VideoComp 800-series machines for many years. This can be done with CRT typesetter's microfilm attachment. The images to be scanned must be reduced to 35-mm microfilm. Scanning takes place by the use of the CRT output tube, which is then used for "input." The data scanned is written to magnetic tape and from mag tape it can be read in as digital information.

Children Back In Style?
How Four Couples Feel

Until recently, millions seemed to be opposed to parenthood—but the trend may be shifting. A sampling of households gives clues to the future.

After years in which childbearing lost its appeal for many American women, the call of motherhood now is beckoning more strongly.

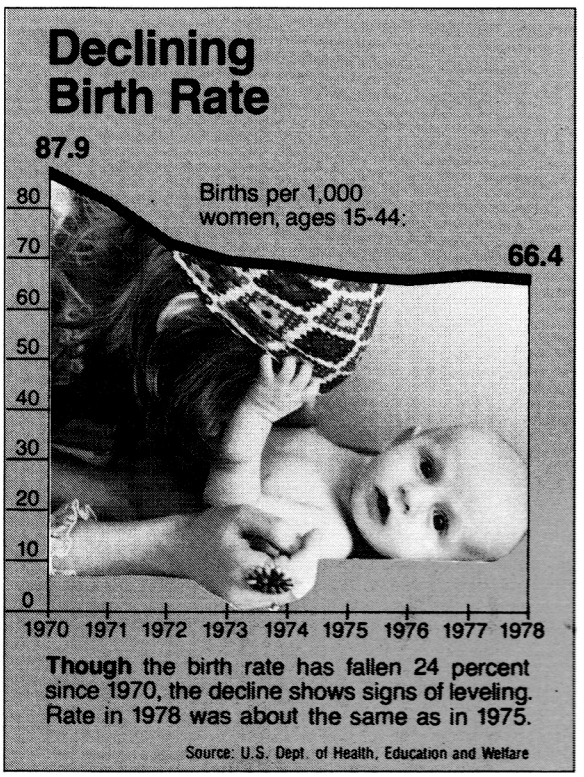
Couples in their early 30s, in particular, are choosing to start families in numbers that surprise the population experts. A significant proportion of women who had put off having children are changing their minds before the clock runs out.

The result is that after a long decline, the country's fertility rate shows signs of leveling off.

Carl Sandburg combined with a family discussion of whether to go to church that finally convinced Barbara Quick, 28, that she was ready to become a mother.

Barbara and husband Bill, 34, had put off having a child until their income exceeded $20,000 because they were "sick of being poor."

That goal comfortably



Halftones, as well as line drawings, have also been composed on the APS-4 and the Harris Fototronic CRT typesetters, but only for trial or demonstration purposes. More recently, Autologic has composed line art and halftones on its APS-5 CRT typesetter from scanned, digitized input.

Not only has the halftone been digitized and typeset by *U.S. News* in the above illustration, but also the graph artwork and even the gray tint.

As was mentioned earlier in this chapter, Hell offers a drum scanner digitizer which, when the writer saw it in use, produced its output on paper tape. A line drawing somewhat larger than 8½ × 11 took two million bytes of storage, and could be composed in approximately 20 minutes.

Of course, digitized logotypes are widely used. Autologic supplies a digitizer for this purpose, as did MGD. The Optronics drum scanner has been purchased by various typesetting manufacturers for the input of font information and for the creation of logotypes.

The ECRM 8400 Autokon camera[1]

The techniques employed by ECRM to input and output line drawings and continuous tone photographs, to perform screening, sizing, contrast, and midtone controls, masking, gray scale reduction or elimination and other similar effects could also be modified or could serve as the point of departure for "fac-

1. ECRM is now a subsidiary of AM International.

simile" transmission—reading at one location and "writing" at another, and even, conceivably, with hardware and software modifications, for typesetting. Work which Professors Schreiber and Troxel performed at M.I.T. on an Associated Press project demonstrated the possibilities of interactive manipulation of pictures. Wire service photos (in this research environment) were stored digitally, retrieved, sized, "windowed"—that is, a portion of the picture is selected for use, enlargement, enhancement, *etc.*—and these manipulations could be made interactively on a TV monitor which, for preliminary consideration, can compress the data, working with less information than will be presented for final output. Interim output is viewed and edited on the TV monitor, and is also output on dry process wire service printers. This output is, of course, good enough to provide continuous tone "copy" for newspaper reproduction.

In sum, then, the technology is here with us now—except for more experimentation with dot structures, and the development of an optimized photographic character repertoire.

Nor does color separation or color rendition present any major technological problem. PDI and others have offered relatively high-cost solutions, but lower-cost approaches are to be anticipated.

Data compression

Another area that is in the forefront of current research is the compression of picture information for storage or transmission. Even storage or transmission of photographic *characters*—as opposed to pixels—involves a great deal of data —especially if a wide range of gray scale levels is desired, and if differently shaped photographic characters are created with identical gray scale values. But, as everyone says, the cost of storage is coming down—and perhaps, also, of transmission. Most compression techniques seek to eliminate the transmission of redundant information (where two photographic characters side by side are identical) or to describe only the differences between one character and the next. Other proprietary techniques claim to achieve an even more substantial reduction in the amount of data, suggesting that ultimately high-quality facsimile transmission (not just of characters but of pixels) may be possible, even though the "character" route seems more promising to us at this point.

Economics

In providing this review, we do not mean to imply that we necessarily believe that it is more economical to generate halftones or even other art work by means of photocomposition than by more conventional techniques which are more manually oriented. It is not expensive to position a screened Velox into the allowed window, or even to double-burn a plate. It is a nuisance, but more elaborate techniques are not necessarily any cheaper. The *U.S. News* case appears to offer dramatic advantages to this news magazine, but this is clearly a somewhat special situation: it involves remote printing, and the problem of getting material to the printer to meet tight deadlines and to avoid expensive overtime on large presses. It is certain that there are savings in the reduction of chartered plane trans-

portation costs and messenger service, to say nothing of press overtime, in that particular situation. *U.S. News* in fact projects a very favorable pay-out, and does not minimize potential editorial advantages. Other news magazines, also printed in several locations, are now following suit.

We have afforded very little space to a consideration of the halftone problem in this presentation because we believe that, as a general rule, economies are not necessarily likely to be forthcoming at this time. They may not occur until the technology encourages going directly to plate, bypassing film altogether. But as it appears, each of the steps that are beginning to be taken—pagination, typesetters capable of composing full pages, digitizing, data compression—all of these developments clearly point the way in the direction of "composing" art and illustrative material as well as type. And so it seems to be time to identify some of the trade-offs involved. (The same problems are also relevant for other methods of replication—such as ink jet or xerographic transfer.)

Problem areas

Looking toward interactive graphics, the writer sees the need to develop techniques for sizing type which is to be created by the raster scan technology and yet to maintain a constant resolution. There is a need to be able to work with non-screened illustrative material interactively, to enlarge it, and to position it, and *then* to screen it. There is a need to be able to transmit data as characters rather than as pixels, or to discover some intermediate mode wherein photographic characters may subsequently be enlarged or reduced without affecting resolution. And above all, there is a need to know more about what the ingredients are which go to make up good plate characteristics to do justice to halftone reproduction.

Review Questions:

1. What are "logotypes," "line art" and "halftones"?
2. How can these be input into a typesetting system?
3. How do analog recording techniques differ from digital ones?
4. Cite an example of a typesetting system which employed analog signals to receive and transmit line art and halftones.
5. What are "gray scale levels" and what is their significance relative to analog-to-digital signal conversion?
6. Describe some of the scanning techniques that might be used to "read" a picture.
7. What are some of the factors governing "resolution" during the scanning process?
8. What do the terms "mil," "pel" and "pixel" mean?
9. Describe the function of "screening" halftones.
10. If halftones are to be scanned, digitized, cropped, sized and then output, at what point in this progression should "electronic screening" ideally take place?
11. Enumerate the scanning/typesetting systems currently available which could output a digitized halftone. Describe these systems briefly, pointing out the differences between them.
12. What is the significance of the "contour" of the dot or cell shapes which make up a halftone?
13. What is the "diffraction theory"?
14. Why is data compression an important element in halftone generation?

23

The Typesetting/ Data Processing Relationship

OF COURSE any computer manipulation for typesetting, such as hyphenation and justification, or pagination, is a form of data processing, but in the typesetting field a distinction is generally drawn between typesetting and data processing applications. The first category relates to one-time typesetting (where the product is composed once and the file will never be re-used), and also to reiterative typesetting. in the latter case, a file is created so that it may provide more than one setting of the same material (although perhaps in different formats), but there is no intention to use the file for purposes other than typesetting.

The second category relates to the creation of a file for purposes other than typesetting. Typesetting may be a by-product, but other uses of the data are more important. Or, while typesetting may be the principal purpose, the file also serves other purposes. Or, even though the object is the typesetting of all products, the products themselves are created by means of computer programs which perform functions unrelated to those required for typesetting.

Obviously, the distinctions we are attempting to draw are arbitrary, and yet most typesetting systems—especially those that are minicomputer-based—do not handle related data processing applications. At the moment, at least, very few vendors offer, as part of a typesetting package, or in addition to it, programs that would sort a file in order to rearrange entries so that they would appear in a different sequence from that in which they were input, or programs that would "explode" a file, to take portions of one entry and list them in various categories. It is probable that vendors of small computer systems will add more and more data processing functions to their bag of tricks, but with the exception of the newspaper classified ad application, generally speaking they have not yet done so.

This does not mean that the hardware they supply is not amenable to being programmed for other applications. Most of the computer systems in use today include a general-purpose programmable computer which, in other incarnations, may perform a variety of tasks which go under the heading of data processing.

But often typesetting and related text editing chores take place in a dedicated environment and—at least while they are going on—no other programs may be run. This would not be true of larger systems, of the IBM 370 variety. In this case it is entirely possible to conceive of typesetting and text editing as being carried forward in one partition of computer memory, or under what is known as a "virtual" system, in conjunction with a wide range of other acitvities. At the moment, however, there are not many composition systems which are being run in a multi-programming, time-shared environment.

It is perhaps more common for minicomputers used for typesetting and text editing to be relieved of their text processing obligations for certain periods of time—such as a third shift—and to be devoted during such hours exclusively to business purposes, such as the generation of payroll, or sales analyses. When this is done it is even possible that the entire hardware configuration is directed by a different operating system.

But our interest in this chapter is not to discuss whether or not typesetting computers can be used for non-typesetting purposes. It is rather to consider a quite unique relationship between certain kinds of relatively conventional data processing activities and text processing, with typesetting as the objective. In other words, we are interested in the interface between non-typesetting and typesetting activities which involve the use of the same basic data.

The typesetting service bureau

During the 1960s and early 1970s, most of the so-called computer typesetting activities (outside of newspapers) grew out of the desire on the part of the ultimate customer to typeset data from existing data bases, or to create data bases while preparing material to be typeset. It was felt, during that period, that "straight typesetting" by computer was not likely to produce costs low enough to compete with conventional means of setting type. But what was true then is not true any longer, and in fact the entire emphasis has shifted to the development of on-line editorial systems in which data bases are not relevant except to provide for the transitory handling of text files during the period between their creation and their ultimate destruction after the newspaper or periodical has been published.

But now the emphasis is again shifting back to a further examination of the value of "data bases" *per se*, and to the relationship between the creation of the data base and the publication of all or some portion of its contents. In order to get a better understanding of what we mean by "data bases," suppose we look at the subject from the point of view of a commercial typesetter in search of markets or applications for his equipment and software. What uses might he find in the "data base" field?

Substituting for the line printer

In the first place, he may discover that there are many occasions when it is desirable to typeset what would otherwise be presented as computer line printer output. If more than five or six copies are required, some superior form of

replication is necessary. Printouts can be photocopied and reduced to printing plates, but type will often do a better job. Type is proportional and takes up less room. For running text, type proportionality will reduce the amount of space required by approximately 40%, assuming the same-sized image. A 40% saving in printing plates, paper, handling, storage costs and mailing may easily justify the effort of "going to type"—especially if this can be done without having to rekey or re-input the data. The primary reason, for example, that the U.S. Government Printing Office ordered the high-speed Linotron 1010's, and the Master Typography Program developed to support them, was to save printing and paper costs. And the savings were sufficient to pay for the expensive equipment in a few years' time.

As the quality of line printer output improves or alternatives such as the daisy wheel, the ink jet or the high-quality electrostatic printer become more ubiquitous, the reasons to "go to type" may, however, become less compelling in the future. But type does lend emphasis and clarity, because varying sizes, weights and faces can be employed.

In any event, many a commercial typesetter with computerized equipment is confronted with the challenge as to whether or not he can reformat data prepared on a line printer so that it can be typeset, or whether he can provide assistance or guidance to his potential customer—the owner of the data base—so that the latter can do the necessary formatting.

If the reformatting is done by the customer, the typesetter-vendor gains only in that he will derive revenue from the sale of photo-unit time. But photo-unit time is what he has the most of, and he is generally delighted to find someone who is prepared to "hang" a tape on his CRT device for a relatively long run, and who will be a source of very little trouble, demanding nothing of keyboarders, proofreaders or programmers.

However, few customers have the programs and the capability to provide output in a form which can be accepted directly by the photo unit. The ones who have developed this capability are customers whose volume is so substantial, and where the economy of this method is so great, that they have taken the time and trouble to create the special-purpose software which their jobs require.

The customer would have to know what code structure a particular typesetter requires, what commands it needs to make it go through its paces, what fonts are available, and what character repertoire is required. The customer might also have to write an "h&j" program, and even a page makeup program suitable to handle his needs. In some cases this might be relatively easy. If he has a data base consisting of a telephone directory or an airline guide, he could avoid some of the major composition programming challenges.

A second alternative would be for the typesetting vendor to "massage" the customer's data, formatting it for his own photo unit. This can be done either by

a generalized program which interfaces with the customer's file structure and data file, or a special-purpose program which performs whatever functions are necessary, often bypassing the composition program, but preparing the data to be handled directly by the photo unit. If the definition of the assignment is to set in type line for line what otherwise would have been composed on the line printer, the task is generally not a difficult one. Hyphenation and justification are not needed and page makeup will be the same as that which previously had been performed on the line printer. A one-time program would handle most jobs of this kind.

It often happens, however, that the customer possesses a data file which he does not understand, since it was developed for him by another supplier, engaged in conventional data processing activity, and he wishes the typesetting vendor to pull from that file only certain data elements and to reformat and perhaps resequence those elements. The typesetting vendor would necessarily have to develop a special program to handle this one task, although he may already have created certain subroutines which he could string together to solve part of the assignment. If he must prepare such programs "from scratch" —and if they are not going to be used very often—he would find it expedient to write them in a "high level" or "compiler" language, such as PL-1. After selecting and manipulating the data, the vendor may then format the run so that it would pass through his composition program, or he may decide to bypass it altogether.

Perhaps there is no special reason why the typesetting vendor, rather than the customer's usual computer center or computer facility, should find himself deeply immersed in data processing activity of this character. On the other hand, he, more than anyone else, knows the input or format requirements of his composition programs, or, alternatively, of his typesetting output device. Moreover, he will have developed a certain facility for character-by-character processing which is somewhat foreign to other computer users, and he should already know quite a bit about his customer's literary and stylistic requirements. For example, conventional sorting and sequencing runs, taken off the shelf from a computer center's supply of utility programs, will often not cope with the somewhat more elaborate rules the customer has developed over the years. Library catalog sequencing, for instance, is highly specialized, and even the mundane telephone book presents some interesting problems. Moreover, these seem to differ from company to company, even within the Bell system.

The opportunity thus presents itself for the computer typesetting establishment to become a specialist in the handling of data for publishing applications. These involve many peculiarities not otherwise encountered in the computer world. Points and picas, extended character sets, widows, telltales, principles of indexing, the ability to arrange and present material economically and artistically —these should be the stock in trade of the new compositor. And those who have managed to learn this trade, and have the necessary hardware and software to practice it, are often referred to as "service bureau typesetters."

General categories of activities

The listing below may serve to indicate the variety of ways in which typesetting and data processing activity can become intertwined:

1. The typesetter merely provides the output device, accepting input from his customer's computer.

2. The typesetter accepts as input a tape formatted for the line printer, and processes it through a special program which prepares a tape for his output device.

3. The typesetter accepts as input a tape containing data in the proper sequence, with the necessary generalized codes embedded at appropriate places in the text stream. He re-formats the tape, translates or interprets the codes, and processes the material through his composition program.

4. The typesetter accepts as input a tape which contains only the data to be typeset, but which requires all codes to be inserted by a program. He prepares this program and processes the input through his composition system.

5. The same as point four except that he develops a special-purpose composition program due to the size of the job or its repetitive nature. This special program will run more economically than the use of his own more generalized composition software.

6. The input supplied to the typesetter contains irrelevant information. The typesetter pulls only the data he requires, and then proceeds as in point four or five.

7. The same as point six except that the typesetter resequences the data he selects from the supplied input.

8. The typesetter creates the original input for typesetting, but pulls from his file and returns to the customer data for use in other computer manipulations.

9. The typesetter *creates the original file* for typesetting and for other purposes. It is a structured file, which permits him to perform such operations as extraction, explosion, sorting, abridging, abstracting, and indexing.

10. The file created by the typesetter is heavily encoded, but it is not structured. It consists merely of a running stream of text. Nevertheless, he is able to extract certain data from the file, manipulate it, and either feed the results back into the composition program or generate other non-typesetting by-products, such as mailing lists or computer printouts. The typesetter becomes the customer's computer center for all purposes related to the data base which he has created for the customer.

Obviously, this listing is only illustrative and there are many combinations and permutations of the points enumerated. For example, from a text stream file the typesetter may generate an index for an unabridged edition. He may then create an abridged version and generate an index from this. Or he may create an index each month from a certain publication that he typesets and then cumulate the indexes quarterly, annually and so on. He may even cumulate the articles in the publication, especially if they are brief abstracts which are most conveniently gathered together in a more logical fashion than the manner in which they first appeared.

Moreover, in all of the examples listed above (except the last two), we have assumed that when data is supplied to the typesetter it will first have been processed by another computer. It is entirely possible, on the other hand, that the typesetter is provided with the initial input, which may or may not have served some other purpose. For example, his customer may have a price list which he maintains on punched cards. He now wishes to typeset this list. Formerly he produced a printout from his unit record equipment. (He had, let us suppose, a punched-card system which antedated the computer.) Or, the customer has keyed his data directly onto a magnetic medium by some sort of key-to-tape or disc device.

Characteristics of data processing activity

Probably the single most relevant characteristic of conventional data processing (as opposed to text processing) is the existence of *structure* to the data. In other words, the information is *fielded,* and the most usual arrangement is for the data to be structured in *fixed fields*.

We shall assume that the reader knows nothing about data processing. The best way to begin his education would be to consider the punched or Hollerith card,[1] since this is the way in which most structured data (and some unstructured input as well) comes into being in the data processing world. The usual punched card contains 80 columns. These columns may be arbitrarily divided into fields. In each column one character may be written. Twelve positions are available in each vertical column. The ten bottom rows represent numbers, 0 through 9. One punched hole per column is thus used to indicate figures. When a hole is also punched in one of the top two rows (or in the zero row), and also in one of the rows 1-9, a non-numeric character is indicated.

1. See page 161 for further elaboration and illustration.

Two significant factors emerge. One is that the character repertorie of conventional data processing, especially from punched card input, is extremely limited. The other is that the position of the data in fixed fields across the card enables the program (either the wired circuit board of unit record equipment, or the software of the computer) to identify each element.

A case study

Let us suppose, for example, that we wish to prepare a listing for a college alumni register, and we wish to present the following information: name, address, class, and department of specialization. We want to be able to produce one listing

in alphabetical order irrespective of class and department. Another listing is desired alphabetically by year of graduation. A geographic listing is also sought, again arranged alphabetically, but by city and state. Finally, perhaps a listing by department of specialization, irrespective of geographical considerations, may be useful, For some listings only the name and class will appear (not the individual's address). We will keep our example simple by presuming a male college, so that women's married names do not complicate the picture (since this would imply a great deal of cross-referencing or duplicate listings), and we will not differentiate between the dropouts (who nevertheless were enrolled in the "Class of '52") and those who graduated. Let's even forget about those who repeated a year, or came back at a later date to finish their education. But these, and other problems, cannot in fact be disregarded, and often are sufficient to make the system as initially conceived entirely worthless.

Since we are going to sort on names, we must consider how this is done. We must have a field designated for the last name, and another for the first and second names. (If there are "Jr.'s" or a presumptuous "III," this will become part of the given name rather than of the surname.)

How long should the field be for the last name? If it is not long enough, it would unfortunately be necessary to "truncate" someone's name, and if the list is ever published or used for a mailing list, alienate him and lose his contribution to the alumni fund.

If we had all names on a computer we could search for the longest name. But we are just starting to create the file. So we ask around and try to learn what others have done. We also scan the printed lists available to us. We finally determine that the longest name of a graduate was Allenby-Whittall (and now we know that we need a hyphen in our character repertoire!), but we decide to allocate 18 columns for the last name field. Although we keypunch only the characters required, from left to right, our program will fill up the remainder of the field with "trailing zeros" in order to create a uniform sort field:

```
AARON0000000000000
AARON0000000000000
ABBOTT000000000000
ALLENBY-WHITTALL00
BEAUCHAMP000000000
```

and so forth.

We decide, then, to allocate twelve colums for the rest of the name. Hence the first name will begin in the nineteenth column:

```
AARON0000000000000JONATHAN ROO
AARON0000000000000LEONARD S JR
ABBOTT000000000000ARTHUR W III
ALLENBY-WHITTALL00PERCIVAL0000
```

and so we are well started on our "system" design.

Now, turning to the problem of addresses, we visualize that we will want to arrange our information by state and city. What do we do about suburbs? Should "Shaker Heights" be listed as a part of "Cleveland"? Perhaps we are better off sorting by zip code, and then playing the tape against a zip code indentification file supplied by the Government. But let us ignore this possibility and try to solve the problem ourselves, and we shall take the lazy way out by deciding to sequence the geographic file by state and then by town, without any effort to group adjacent towns. Hence we decide to "field" the address portion of the entry as follows:

House number and street
Town
State
Zip code

Let us consider the problem of *state* first: the easiest way to represent the state is by a two-digit alpha field: *MA* for Massachusetts, *CA* for California, *NJ* for New Jersey, *etc.* However, if we are going to sort on these abbreviations, we will get our states out of order. *NC* for North Carolina will sort ahead of *NJ* for New Jersey, for example. Or, we could number the states sequentially. If the file is only to be used for typesetting, these numbers could be stored in a "table look-up" and the name of the state could be generated. We can do the same thing for the computer printout run for mailing labels. It is only if we want a card print that the initials or abbreviations need to be used. On the other hand, it is easier to use the alpha abbreviations than to have the input operator perform a look-up operation in order to assign state code numbers. So we settle for a two-digit alpha representation and we assign to our computer program the task of sorting these in the proper order at a later stage of the game.

So far as town or city goes, we can truncate the town if we like, in case it is too long, and again use a computer program to generate the full and complete name, if this is desirable for reasons of clarity or taste. It might reduce keyboarding if we told the operator to input only the consonants and use a table look-up for all names. On the other hand, it might result in creating some redundancies. Moreover, a listing of many hundreds of towns would be necessary in order to interpret the input mnemonics. So we conclude that we had better keyboard the town's entire name to the extent that it can be represented by 15 columns. We tell the keypunch operator that if it cannot be so represented, to truncate at 15, and write down these names so that we can expand them in our computer program. This will reduce appreciably the number of exceptions we shall have to contend with.

We do not anticipate having the need to do anything special with the house number and street field, other than to print or typeset it, but we must allow adequate room for it.

Three other fields remain. We must designate the graduating class for each person listed, and we can do this by two numbers, such as *60*. (We can assume

that all numbers referring to years yet to come actually represent those years in the preceding century, such as 1895.) The next field will indicate the major subject of the alumnus. One digit will suffice, since we are pressed for space and we can indicate 36 different possibilities by using a letter or a number in this column. Finally we wish to indicate whether the degree conferred is an A.B., a B.S., or perhaps a Ph.D., D.Litt., or M.D. Each degree can be given a number, and *1* may as well stand for the A.B.

By careful planning we have just reached the allowable 80 columns, and we can avoid the necessity of having to overflow onto a second card. Here are the allocations:

1-18	last name
19-30	first name
31-54	street address
55-69	city
70-71	state
72-76	zip code
77-78	class
79	field of specialization
80	degree

It will now be possible for the punched card itself to serve as input to a print format program which would generate labels. Additional programs can provide geographic and class listings, or could produce a sort by kind of degree or major subject.

The master file can be kept on punched cards, and when addresses are changed, the cards can be physically removed and new ones put into place. Or the file can be maintained on magnetic tape or disc, in which case a file update program will need to be written or adapted.

Our typical entry will read as follows:

```
AARON0000000000000JONATHAN R00APT C 17698
MACARTHUR AVWINSTON SALEM00NC60310
```

Going into type

Let us suppose, now, that we are given an assignment of typesetting the alumni directory from these punched cards, or from a tape image of the cards. The first listing we are to produce is to be the master list in the alphabetic sequence in which the file is maintained. Our program would work something like this:

1. Read in a dummy entry which will set up the job, defining the line length and giving the program other data it may require.

2. Read in the first regular entry and store it in core.

3. Take the first field and move it to a new location, dropping the trailing zeros, and inserting, in front of it, a flag which will subsequently identify the type face. Follow the field with a comma.

4. Read in the second field, dropping trailing zeros, insert another typographic flag if the type face is to be different. Follow the field with a period.

5. Then go to positions 77 and 78, put in a flag to indicate the desired type face, then an apostrophe (as in '90) and follow the two digits with a comma.

6. Then go to position 79. Do a compare or table look-up which indicates that because the value in this column is *3*, the letters *Econ* should be substituted, with a flag to designate italic.

7. From column 80, the number *1* will be translated into A.B. followed by a quad left.

8. Then go back to position 31 and pick up the string from 31 to 54, dropping any trailing zeros, and preceding the entry by an em space, to cause the address to indent. A typographic flag code may also be added. Note there is no punctuation in this field. We may wish to interrogate this field for certain peculiar conditions. We will return to this point later.

9. Then follow directly with the name of the city, beginning with position 55. Perhaps this name has been truncated. (We hope not!) But we may require a routine to check first to see if there are any trailing zeros in the field. If not, the name may have been truncated. The name would then have to be compared against a list of truncations, and if found, then the full string of characters spelling the entire name would be picked up. Preceding the city name would be a flag for the desired type face, if different. A comma follows the end of the field.

10. Then proceed to columns 70 and 71. Let's make it easy and assume we are merely going to pick up the two letters we find there. We recognize that we have not placed a period after the *N* in *NC*, and so presumably we decide not to put a period after the *C* or after *CA*, so we insert a comma, and follow with field 72-76, where we pick up the zip code, and end with a quad left, and then a blank line before going to the next entry.

11. We write this information to magnetic tape or disc, and it is now, hopefully, in the format required to enter our composition program. Perhaps we have also inserted other flags to indicate the beginning and ending of an entry so that our pagination program will not break entries when it divides the material into columns and pages.

Although this description seems complicated, the task has been a relatively simple one. But since we also are to be required to produce a listing by state and town, we might have given consideration to creating the information we needed

for this purpose at the same time that we prepared the composition input tape for our main listing. In the interest of clarity, however, let us assume that we regard this as a separate and distinct processing run.

We return, then, to the initial file, and we recognize that we must create a sort field. First we wish to sort by state, and then by town. We know that the number of states (fortunately there are no foreign countries!) is less than 99. We call in the state field and assign a number. A translate table tells us that *CA* is California, and that California is the fifth state in the alphabetic sequence. And so we write out *05*. We follow this with the name of the town, allowing fifteen places, and then we pick up the last name of the alumnus, since we also wish to sort alphabetically by last name where there are several persons listed as residing in the same city. We must also consider the possibility that two persons with the same last name will live in the same town, and we need to pick up, as well, the first name of the alumnus. And so we have created a sort field as follows:

```
05WHITTIER0000000ALLENBY-WHITTALL00PERCIVAL0000
```

In addition to creating this sort field, we will need to develop a "trailer" field which will contain the information we wish to typeset. In this case all we wish is the name and class of the individual. Let us suppose, although this is unlikely, that we decide to set the first name first, so that under California, and under Whittier, we will find:

```
PERCIVAL ALLENBY-WHITTALL, '59
```

There is no problem in doing this. The sequencing will be by last name, but the listing will begin with the first name. The only difficulty we will encounter is that we may have instances like this:

```
LEONARD S JR AARON
ARTHUR W III ABBOTT
```

because we did not take the trouble to "field" these post-surname characteristics. (If we had, we would not have been able to "run in" this information in the same field, and more space on the fixed field card would have been required.)

At any event, after we had sequenced our entries by the use of the sort field, the program would drop it and pick up only the "trailer" which contains those fields of typesetting information we require, in the sequence in which they have been arranged, and with the appropriate "before and after" information around them. By "before and after" information we mean such things as flags to indicate type faces or format instructions, and repetitive information, such as commas or dashes to separate one element from another, as well as to separate one entry from another.

That is all there is to it. One thing should be clear: it is easier to develop such a program if there has been communication from the very beginning about the

typesetting possibilities of the input, rather than to try to improvise from a file which was created without such uses in mind.

Consider, for example, the problem of setting the same file in upper- and lower-case. The customer proposes that this be done, since it will be more legible and will take up less space. An investigation of the name field reveals certain peculiarities, however. The solution is not so simple as to pull off the first letter of the last name, make it a capital, and then to convert all of the other letters in that last name to lower case. We encounter *McAdoo* and *MacDonald*, but we also find *Mackey*. There is a *Dupuy* in the class of '42 and a *duPont* in the class of '98. The latter, by the way, is a substantial contributor! Some of these problems can be solved by special routines, but now we are obliged to carry on a different level of data processing. Until now all we have been concerned with is to place characters at the beginning or the ending of identifiable fields (or in some cases to translate a field from one thing to another). Now, unfortunately, we have to make an analysis of the contents of the field itself. We can write a program to compare for *Mc* and can probably get away with the capitalization of the next letter. We can gamble with *Mac*, although the consequences will be somewhat more treacherous. We'd better give up on the *du* and *de* problem. An editing pass may be necessary to get the file into shape for typesetting. But if we do this, can the file be used again for other purposes, such as the generation of a mailing list? Or will it be necessary to edit the file each time a typesetting application is planned? These are difficult system questions to answer. Again, prior communication is indicated. If a dollar sign could have been placed ahead of the exceptions, the problem could have been resolved nicely:

```
MAC$DONALD
$DU$PONT
```

which tells us to reverse the normal capitalization rule, but this takes up more room in our fixed field areas, and these irrelevant marks may interfere with the use of the file for mailing lists or for sorting, unless we incorporate into these programs some procedure which causes the routine to ignore them.

If it is difficult to infer the proper position of upper case letters in the name field, it is all the more perplexing in the address field. The rule that the first letter in the word should be capitalized would produce *Afb* rather than *AFB* (for Air Force Base), for example.

Creating a typesetting file

Just as he often works with fixed field data supplied to him by the customer, so our computer typesetter may be asked to assist in the development of the initial file, and he will possibly choose to use some other method of "fielding" the information. He may elect to use variable length fields, and he may decide to place these in random order, or even merely as a part of a text stream. But let us take a case where the sequence of elements would be prescribed. Suppose, for

example, he wishes to create a library catalog. He may develop a concept which permits the entry to look like this:

```
$aCooper$bJames Fenimore$cThe Last of the
Mohicans$dMacmillan$e 1896, 395p.$f$gF$h$i$j$k$z
```

We have flagged the author's last name between *$a* and *$b*. We have flagged his given names between *$b* and *$c*. Thus we can find these elements and create sort fields when necessary. We can also find them when we wish to pull them out for typesetting.

The title of the book appears between *$c* and *$d*, and a program can be written to drop off the articles *(The, A, An)* when a title sort is indicated, in which case the next word should begin with a capital if it does not already do so.

The publisher appears between *$d* and *$e*. Miscellaneous information which we shall typeset but not use for sorting, and which, when typeset, will always appear together, is found between *$e* and *$f*. The identification of the price of the book appears between *$f* and *$g*. The identification of the category of book appears between *$g* and *$h*, and in this instance the *F* stands for fiction. Thus we can sort all fiction together, or identify the book for the reader by Dewey number or LC number, and sequence on these fields. Finally, we have indicated that this book can be found in three libraries in the system, which we have chosen to identify uniquely by the codes *$i*, *$j* and *$k*. The entry ends with the flag *$z*.

This is a simple application. Library catalogs are of course much more complicated than this, since they also deal with subject indexes, but the example suffices to make our point. It is now necessary to have a program which responds to commands to create sort fields and trailer fields, generates information before and after the fields in the trailer, and otherwise manipulates the entry much as in the case of the preceding illustration (from the alumni directory). But since the file was initially input for typesetting, upper and lower case codes, and other internal flags, to extend the character repertoire, *etc.*, can be widely used, so long as they do not interfere with the sorting rules. Sorting rules themselves can be made more sophisticated by intermixing "file as if" commands where normal sorting sequences do not produce desired results.

Other opportunities and problems

For sorting purposes, variable length fields must be converted into fixed fields, so that apples can be compared with apples. But for other purposes, variable length or random fields are much more convenient. It is not necessary to structure the input in a rigid fashion.

In fact, we might consider this type of input to be non-structured. The concept of "structure" in this sense would mean that the item would be identified by its sequence or its position or both. Naturally, there can be some element of

structure even in a random file, and wherever structure exists certain conditions can be inputed to the data, so long as a beginning and ending can be determined.

Flags can be put around names which occur in random order in the text stream so that these names can be recognized for the purpose of creating an index. Material to be omitted for an abridged version can also be identified. First lines of poetry can be tagged so that an index of first lines can be generated. These are only a few illustrations which come to mind. Space does not permit a further elaboration of this topic, which is obviously deserving of an entire book in its own right. But perhaps enough has been said to start the reader thinking about the ways in which typesetting and data processing can be interrelated. Since publishing need no longer be considered exclusively a function of typesetting and printing, it is obvious that many publishers will wish to store their products in such a way that they can provide information through a number of media, including a magnetic tape file which others can search or manipulate for their own purposes. Information retrieval and the selective dissemination of information is also possible from files which have been input with these applications in mind. The more thought that goes into the development of the original input, the greater the potential for by-products and alternate uses.

Definition of data manipulation terms

Some forms of data manipulation often used in typesetting applications include:

- **Extraction**—Selecting only certain elements from the text, such as the name of an author, or a citation or title.
- **Explosion**—generating the same information in different arrangements from the same initial input. For example, the name of a company followed by the products it produces, and then creating a list of products with the company name restated under each product.
- **Abridging**—eliminating certain material from the text stream.
- **Abstracting**—summarizing the essence of an entry or article.
- **Automatic indexing**—generating a listing of index terms in alphabetic sequence, with reference to the page numbers where the item occurs. The index terms are selected by computer program, such as to pick out all words in the text stream of five letters or more in length, and assuming that words so used with frequency represent key words in the text.
- **Redundancy elimination**—dropping a portion of an entry when it already appears in a preceding context. For example:

U.S. History—Colonial Period—Early Settlers, p. 67
—Handicrafts, p. 14
—Revolutionary War—Anecdotes and Satires, p. 4

Programming implications

While it is true that the use of the file for other purposes makes computer typesetting much more viable, economically (and everyone these days wants to find himself a unique data bank which he can exploit on the market place!), programming and systems design are complex and far from trouble-free. Computer typesetters who have forsaken straight one-time setting for data bank applications, on the ground that they cannot make a profit in competition with conventional typesetting, also find that the data processing path is a thorny one. Programming ingenuity is obviously necessary, plus the ability to anticipate the customer's requirements and point out the pitfalls before they are encountered. These skills are rare, and their absence keeps the typesetter from profiting by the opportunities he is able to uncover. Moreover, neither he nor the publisher may possess the insight to discover the wealth of applications which lie around them.

Illustrations of data bases

Some of the obvious data bases are listed below:

1. Telephone directories, especially including automatic updating, cumulating, reverse directories (by street address) and recombinations of elements within the directory, such as by exchange, or businesses, or types of businesses. Especially interesting are applications where the input can be derived from business files, such a service orders.

2. Airline schedules, where information is computerized in any event, and points of interconnection must be retrieved because of their effects on the operation of the total scheduling system.

3. Library catalogs, especially where the information can be derived from computerized book ordering systems within the library.

4. Publisher's book catalogs, especially where various combinations (*e.g.*, series) are published, and the same listings may be needed for inventory control.

5. Product guides and industrial directories, especially where information can be elicited from computer-generated questionnaires.

6. Price lists and parts lists, especially where information is maintained for inventory or billing purposes.

7. Abstracts and bibliographic citations, especially where cumulations, sorts and explosions are relevant, and where information can be input during the process of intellectual abstracting.

8. Tariff listings, stock market transactions, bond holdings and other collections of data which require widespread dissemination and constant revision.

From a typesetting standpoint, there are certain unique features to the above classifications:

a.) Input costs are not usually relevant. The information has to be created and maintained in any event.
b.) Updating, cumulation and other data manipulation processes are common.
c.) The information is usually quite massive and repetitive.
d.) Typographic or composition demands are often quite simple. Formats, once evolved, do not vary. Special-purpose high-speed composition packages are logical.

The editorial interface

Most of the data bases we have so far alluded to are structured, either in a fixed-field or a variable length record fashion. But there are also data bases which are not structured at all. The Holy Bible (*e.g.*, King James version) is a data base in that it can be typeset with or without footnotes, or with different levels of footnoting; commentary, such as chapter headings, can be present or absent; different typographical formats and styles can be applied. Moreover, proper names in the Bible can be searched and verified for common spelling and uniform hyphenation.

Similarly, insurance forms and contracts, as well as "boiler plate" for employee booklets, can be stored. Programs can be written to extract the relevant paragraphs or provisions and put them together in order to meet the requirements of different policies and the laws of different states.

Statutes for various states need to be typeset. The same data base could provide a source for annotation. And legislative bill drafting programs can be offered whereby anyone wishing to submit a bill for legislative consideration can call up the existing law, and indicate the manner in which it is proposed to amend or revise it. The drafts, as revised by the House and Senate committees, can be changed as required, and a trail can be provided showing what revisions were indicated and at what stage in the process. The final bill, as passed, can then be re-inserted in the new revised statutes.

These are merely a few illustrations to suggest that the computer may be called upon to perform certain "editorial" functions—that is, tasks which were formerly done by people who worked tediously, usually reading and copying portions of text onto file cards, inserting or deleting from earlier versions, and searching for other occurrences of the same conditions.

More and more attention is now being focused on the development of large data bases, in which relevant portions can be retrieved upon demand. Often the text to be searched by the computer is unstructured. A newspaper, creating text for its news stories, may wish to store the text for future reference, in a computerized morgue. Programs may be required to search these files for names, or places, or phrases. Or, an abbreviated version of the text may be stored simply for the

purpose of permitting the kind of computer search which would lead the user to the appropriate issue of the paper and perhaps automatically retrieve the microfilm display of the page or pages in question.

As the cost of data storage is reduced, and as computers become faster and cheaper, more and more interest is being displayed in the use of text-searching techniques as an aid to research and scholarship.

In the future, the distinction between textual material and "data bases" will become less and less significant. All text that is input will be regarded as a potential data base, and all data that is stored and manipulated will be available to be output by a typesetting device merely as another form of presentation. Thus the typesetting/data processing relationship will someday become so intimate that the disciplinary, programmatic and structural barriers will seem never to have existed.

Review Questions:

1. What is a "data base"?
2. What are some of the areas in which data base maintenance (or manipulation) of typesetting converge?
3. What is meant by "exploding" a file?
4. Enumerate some of the problems of encoding, formatting, sorting or otherwise manipulating data retrieved from a customer-created data base, which might be encountered by a computer typesetting service bureau.
5. What is meant by "fielded data"? What are "fixed fields" and "variable length fields"?
6. What are "table look-ups" and what are some typical uses for them?
7. What is a "sort field"? What information might be contained in a "trailer field"?
8. Imagine that you are a member of a computer typesetting service bureau and are given the opportunity to work with one of your customers in setting up the initial formatting of his data base. Select a sample data base application and describe the manner in which you would want the data encoded for input into your computer typesetting system.
9. Using the same data base example you selected above, enumerate every possible format you can think of in which that data, or portions thereof, might appear.
10. Compare your answers to questions 8 and 9 above. Did you allow enough leeway in your original encoding of the input file to accommodate all of the output possibilities you delineated?

24

Putting the System Together

A BEWILDERING VARIETY of equipment and solutions is available to the potential user who has word processing, text editing, or typesetting requirements. If we conceive that some of the systems and tools which we have discussed in this book are possibly relevant for the office—to replace the typewriter—perhaps the "market" might consist of several million installations, or a lesser number capable of turning out the equivalent volume of productive work. But even if we ignore the office market, it is variously estimated that in the United States and Canada alone there are from thirty to sixty thousand so-called "in-plant" users of offset duplicators for whom some kind of typesetting capability would be useful. It is this in-plant market—including government agencies at all levels, and educational institutions, as well as private establishments—which is currently receiving the attention of such manufacturers of "direct entry" typesetting equipment as Itek, VariTyper, Compugraphic and Mergenthaler.

The next largest number of users or potential users—a market segment which is the most homogeneous of all—is to be found among the daily newspapers. Even here there are wide differences in requirements. Many of the larger papers—those located in the major metropolitan centers—are still printed by letterpress, while the newspapers which service the small cities and towns scattered across the United States now almost invariably use the offset process. Until recently, newspapers printed by letterpress were slower to accommodate to new composition technology. This is no longer the case. Moreover, the overseas market is not to be ignored.

Within the newspaper market place we have seen the gradual introduction of photocomposition equipment. Then came optical character recognition devices. Then, with a passion pursued with singular unanimity by virtually all newspapers of whatever size and shape, came the introduction of editorial systems, with the use of which stories are created, revised, routed and finally composed—all on-line. Meanwhile, classified ads are received, entered, manipulated and published.

Significant problems still remain when it comes to devising an appropriate system for the newspaper industry, quite apart from determining the necessary

size and features in keeping with the requirements of a particular paper. The uncertainties which prevail have to do with the manner in which page makeup decisions will be reached and implemented, and the length of time that will pass before it is feasible for illustrative matter to be manipulated interactively as part of the page makeup process.

Other kinds of uncertainties still prevail in the newspaper industry because, as has often been stated, "the walls are falling down" around the conventional jurisdictional barriers and hegemonies. These barriers are not merely those constructed by claims staked out and honored in the unionization process. Intellectual or conceptual barriers—usually institutionalized by organizational arrangements—are certainly more profound and far-reaching.

As these barriers crumble, new techniques are emerging in response not only to human capabilities, interests, and the force of personalities, but also in an effort to shape, or at least respond to, changes in the market for the final product —the newspaper itself.

For example, at least some newspaper publishers seriously believe that it will not be many years before our newspaper reading is performed at the television console, with the assistance of a printout device to capture material which the viewer may believe to be worth saving for at least a few days. Already several ventures of this kind have come into being.

While typesetting technology will obviously be impacted by such developments, the newly-emerging editing systems will not be found to be anachronistic. Attention will be focused, however, on the storage, selection and retrieval of information for the convenience of readers with varied tastes and interests.

Other publishers

Turning now to the segment of the market or potential market which still, like the newspaper, consists of publishers rather than vendors of composition, we find that the editorial convenience of the new systems is now involving many publishers who, until recently, have bought their typesetting from conventional suppliers, with whom they maintained a more-or-less "arm's length" relationship. Although these suppliers have, of course, been much more knowledgeable with respect to typesetting aesthetics and production requirements than most of the publishers themselves, the latter have found that the convenience of a more intimate relationship between the editorial and typesetting functions is worth the learning experience which appears to be indicated.

The publishers that we are considering are those who engage writers to work on site, and editors, or abstractors, or indexers, or combinations of these, who also work on the premises or in close proximity. Quite often the relationship between those who create the raw material and those who hone it for publication is much more complex than is true in a newspaper environment. However, in most cases it still seems advantageous to seek to capture the keystrokes them-

selves when they are first laid down, whether by writer or secretary, and thereafter to do as much of the editing and revising as is required "within the system."

What that system should be will, of course, be determined by the size and nature of the individual publisher's requirements. These have to do not only with the volume of work to be performed, but also with the complexity of the material. Is it, for example, heavily scientific? The structure and complexity of the required formats and the possibility of creating short-cuts and systems standards needs to be studied carefully. Even highly complex material presents no significant problems if it is susceptible to reiterative use and shorthand methods of invoking these formats can be devised.

One must also take into consideration the extent to which rewriting for copy-fitting purposes is allowed, encouraged or mandated.

The possibility of alternative uses of the same data base, and, consequently, any possible need for long-term storage must also be planned for from the very outset.

Here are a few possible models:

1. Volume small, production cycle or turnaround time not excessively compelling, much of the material received from outside writers, and not too heavily edited, text quite straightforward and page makeup relatively simple.

   This could provide a good case for the installation of a CompuWriter, EditWriter, Comp/Set, Comp/Edit, Linocomp, Linoterm, AlphaComp, Quadritek, or a similar device. An editing terminal—even one that uses paper tape or that works from a floppy disc or cassette, would be most helpful. A stabilization processor would probably be sufficient for the output chemistry, although the more recently introduced dry silver process could be even more attractive. Only the application of heat, by means of an inexpensive transport which can even be built into the machine, as with the Compugraphic 7500 "Dry" model, is necessary to develop and "fix" the photo medium. No significant archival storage for subsequent use is implied.

2. Same conditions as above except that the material is much more complex, involving, perhaps, foreign languages, mathematical equations, and the like.

   In this case if the volume of production is fairly low, the use of an outside vendor would appear to offer the best solution. If, however, vendor charges are too high to be cost-justified for the application in question, the publisher might attempt to explore as an alternative the installation of a minicomputer-based system from the one or two suppliers who presently have developed software sufficient for these purposes. At the moment, mathematical equations appear to present the most formidable obstacle.

3. The volume of production generated "in-house" becomes larger, or the amount of editorial work performed on the manuscript becomes greater, or the time interval between writing and production becomes shorter, and the extent to which copy fitting for presentation is desired becomes more significant.

   Obviously the argument for the publisher to install his own internal capability becomes compelling.

4. There is, in addition, the prospect of using the same information for the generation of other products.

   Here, again, the publisher is led to install his own system, or at least to make long-term arrangements with a handful of "service bureau" typesetters who have developed programs, knowledge and skill in the application of "data base publishing" capabilities.

The "commercial" typesetter

In the commercial area, there are advertising typographers, trade book setters, financial printers, and general trade shops, as well as some few who specialize in foreign language work, or mathematical and scientific journals. The needs of these shops tend to differ widely with respect to equipment and systems. However, there is an increasing tendency for these heretofore highly-specialized enterprises to invade one another's territories since their capacity tends to expand, and the systems or machines which they install prove to be perhaps more useful for the performance of work quite different from that which was intended at the time of purchase and installation.

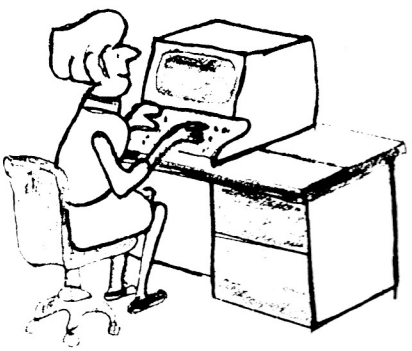
Equipment decisions are more difficult for "commercial" shops than for others, since they cannot tailor their systems to the particular requirements of a given publisher or application, but must perforce respond to more generalized demands.

Hence, it seems to follow that almost anyone who has a specialized and well-defined need will do better to "do his own thing" than to rely upon a conventional trade supplier. The latter is left to handle all of the odds and ends of work which remain. This is neither surprising nor unusual, however, since the trade house has usually been obliged to respond to a wider range of requirements—especially when it comes to type faces—than the in-house type shops.

Nevertheless, it implies that, by virtue of this diversity, the trade house, in trying to be all things to all people, may do few of them as effectively as they might otherwise be performed. Trade shops will be hard pressed to select equipment to span the entire range of services that they would like to offer. They may not be able to afford such an ambitious charter.

Some representative concepts

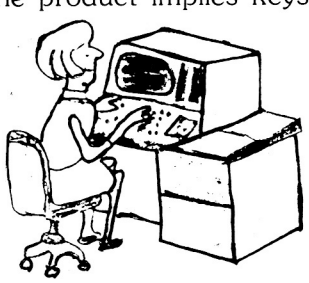
The task of defining one's role and one's requirements is not an easy one. To suggest some of the alternatives it may be helpful to review the system models which follow. The reader may wish to ask himself where these models would seem to fit, and what markets or uses they would address most successfully. To assist in performing this exercise, we have indicated a range of prices associated with the equipment or systems which are presented. We cannot (in a general book of this character, which is intended to provide an overview) linger on the pros and cons, or the detailed features of any of the alternatives we will now lay out. But these diagrams may help the reader to examine possible systems or "solutions," and to identify the concepts which these systems most closely approximate.



I. Direct-entry products and concepts

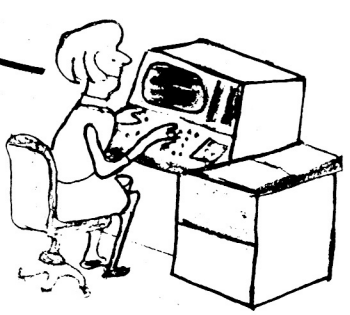
Direct-entry typesetters imply that they are one-operator machines. The operator keys the input into a buffer from whence it flows directly to the typesetter-portion of the same machine. It may be run in a counting mode, whereby the operator makes the end-of-line decisions, or in an automatic or semi-automatic mode, wherein some or all of the end-of-line decisions are made without operator involvement. Some portion of the input can be accessed while in the buffer, prior to typesetting. Such stand-alone devices usually sell in the price range of $6,000 to $12,000, depending upon the speed, number of fonts and sizes on line, and other features. (Examples: CompuWriter, Comp/Set, EditWriter, Quadritek, Linocomp, AlphaComp.)

The next iteration of the product implies keystroke capture on magnetic cassette, paper tape, or, most likely, floppy disc. The act of input may be separated from the act of typesetting, by storing the keystrokes in an unformatted or formatted data file; or, this file may be created while typesetting occurs. Various file management schemes are offered to help keep track of work in process, and to facilitate the editing and composition functions. (Examples: EditWriter, Quadritek, Comp/Set, Comp/Edit, Linoterm, AlphaComp, VariSetter.) Prices of such "systems" would begin at around $12,000.

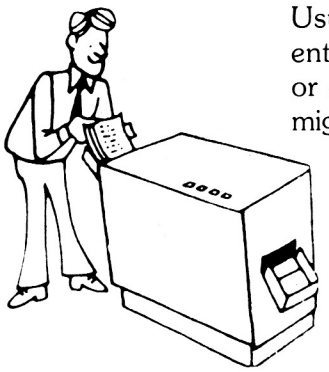




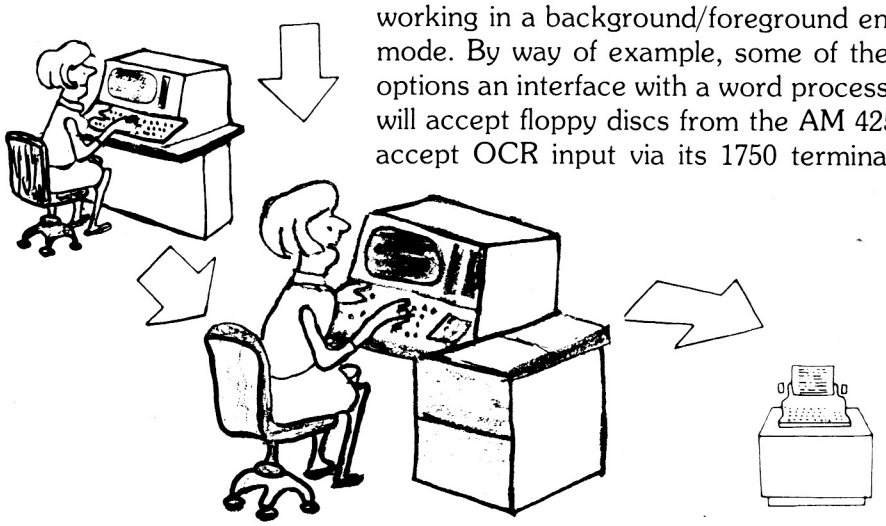
The next logical progression permits off-line input, so that the "direct-entry typesetter" may still be used as originally intended, but text would also be keyed off-line on a similar device without the typesetter portion, and the input—formatted or otherwise—transferred to the direct-entry device for setting, usually by mounting a floppy



disc in the typesetter's disc reader. Some such devices can be operated in a foreground/background mode, with one job being typeset on the "direct-entry" machine while its keyboard is used for input and/or editing of another task. Usually, the typesetter portion of the machine runs faster than a simple direct-entry device. (Examples: same as those listed above—perhaps the same models, or other models, or options added.) With a second keyboard the "system" price might still start at less than $16,000.

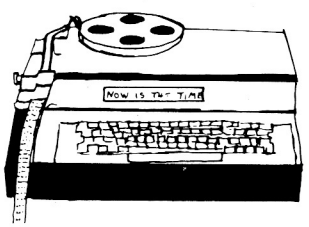


The next step in the progression would be to be able to accept input from some external source foreign to the system, such as OCR, or input from a word processor, a key-to-disc terminal cluster, or even from a paper tape perforator. Such input might be received on line, or by mounting an interim floppy disc in the typesetter's disc controller (or, of course, to use a paper tape reader.) Some such devices can accept input while permitting the "direct-entry" operator to keyboard or edit other jobs, thus working in a background/foreground environment, or even in a multi-tasking mode. By way of example, some of the machines mentioned above offer as options an interface with a word processor or an OCR device. The Comp/Edit will accept floppy discs from the AM 425 word processor. The EditWriter will accept OCR input via its 1750 terminal.



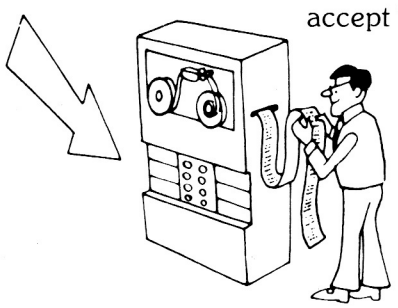
It might even be possible to produce alternate output on a line printer or a correspondence-quality (daisy wheel) device. Bobst/Varisystems' VariSetter might be programmed to offer this capability. One thus might build such a system, adding more terminals and output devices, until it costs as much as $50,000.

The final, logical step, would be to disassociate the typesetter from the entire process, in which case you no longer have a "direct-entry" system.



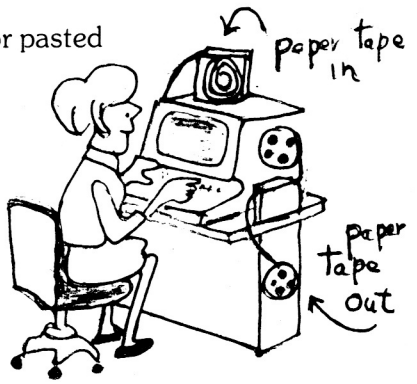
II. Stand-alone Typesetter Products and Concepts

Very few of the second-generation typesetters are "slaves," although when they are used as output devices for "systems" they may be driven without recourse to their own "intelligence." Even many third-generation typesetters can accept "idiot" or unformatted input. The level of intelligence may vary significantly from one machine to another but most of them are able to make end-of-line decisions with some form of logic hyphenation (and perhaps a very modest exception word dictionary). And most offer the convenience of stored formats.



Consequently, it is possible to purchase stand-alone input terminals, usually of the paper-tape variety, feed the paper tape into the typesetter along with the codes which the typesetter expects, and

set type. If there are errors, these lines can be set again and mortised in or pasted over the offending lines.

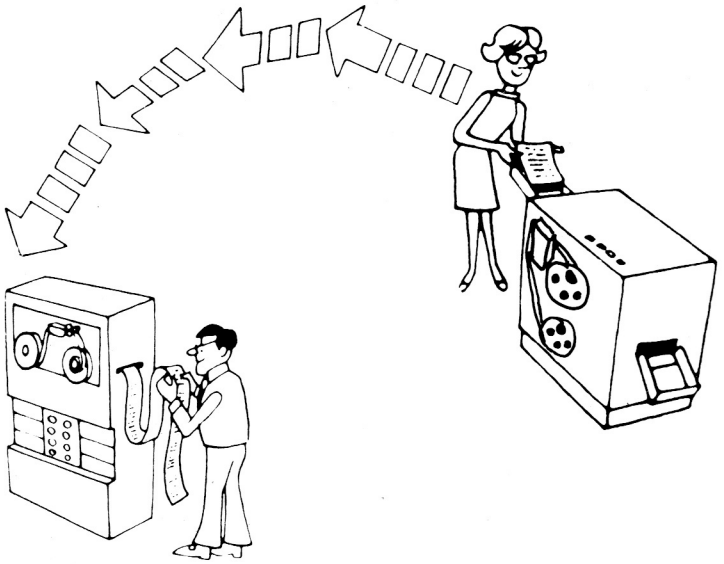


Or the paper tape could be read back through the input keyboard/terminal, assuming that it also had editing capabilities, and corrections could be made, punching out a clean paper tape for resetting of the entire take. If the initial keyboard did not have editing capabilities, such as a paper tape reader as well as a paper tape punch, and a VDT screen with overstrike, insert and other similar features, one could use a stand-alone correction or editing terminal at this stage.

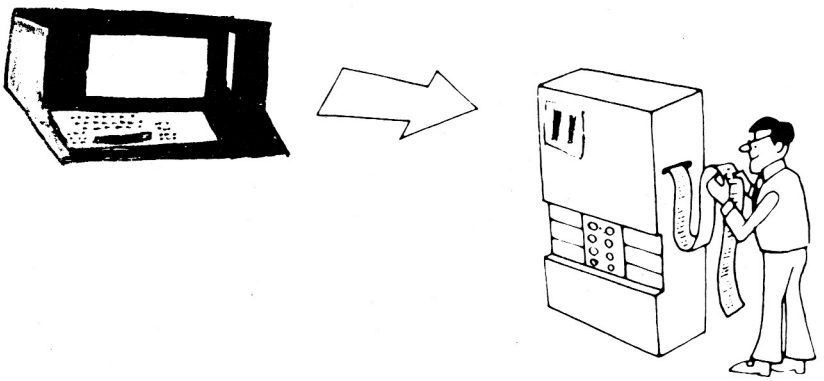
If the typesetter had a cassette reader or a floppy disc reader, output could be similarly prepared, only onto cassette or floppy disc rather than paper tape.

The next step would be to use one or more "counting" keyboards. Perhaps some input could be prepared on non-counting boards, and corrections set on counting keyboards, along with the more complex work. Or the same keyboard could be operated in either a counting or non-counting mode. And, usually, the same keyboard could also be used for editing or correction purposes. One might therefore proofread directly from the VDT screen, and use global substitutions to make changes in improper formats.

In all of the situations we have discussed so far under this heading, one has the option of using keyboard-generated formats rather than the formats which could be stored directly in the memories of the typesetting devices. More sophisticated users will take advantage of both capabilities: perhaps formats stored at the keyboard or editing terminal level will call out other formats stored within the typesetter.



Another possibility would be to produce input via OCR, writing it out to paper tape or floppy disc. This output could serve as input to the typesetter (via paper tape reader or on-line floppy disc reader.) In this event, the post-recognition logic of the OCR device could handle some of the codes and formatting. Even aesthetic kerning could be added to the OCR output stream, to cause the typesetting device to take these "minus" values into account (although some second-generation typesetters, such as the ATP 1/54 software version of the Mergenthaler V-I-P, can do "their own" kerning.) Corrections could be handled by going back to the manuscript input and entering them between the lines, or by cross-outs, and then recreating some or all of the input for the typesetting machine.



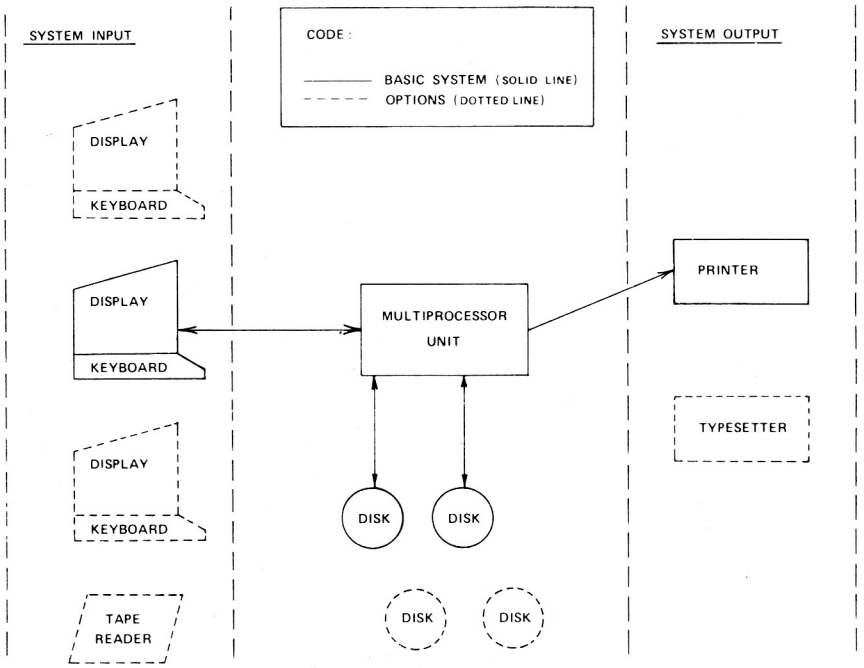
Finally, there are some typesetting machines which can take certain kinds of word processor output and set type directly, using the typesetter's intelligence to provide the end-of-line decision-making. Here corrections might be handled either by stripping or by going back to the word processor, which has its own means of editing the text and into which codes and formats can be inserted.

III. Small system concepts

There is no clear distinction between a stand-alone terminal (which has its own counting and editing capability) and a typesetting "system" as such. (Of course, in the broadest sense, every method of setting type implies that the user has a "system concept.") But the term "system" usually implies several pieces of equipment which are linked together not only functionally, but physically as well. In this sense a "stand-alone" terminal is not a system. But if it were physically interfaced to a typesetter, presumably it would take on the aspects of a "system." Yet if you output to a floppy disc, and then pass that disc on to another floppy disc reader which is connected to the typesetter, there may in fact be relatively little difference in how the work gets done.

A more powerful capability is introduced when a "stand-alone" terminal is capable of multi-tasking or multi-processing. In order to perform more than one task at a time such a terminal has to be on line to something—whether it be at the input end, with a device such as an OCR reader, or at the output end, with hard copy printers and typesetters. Or, of course, both input and output devices may be interfaced.

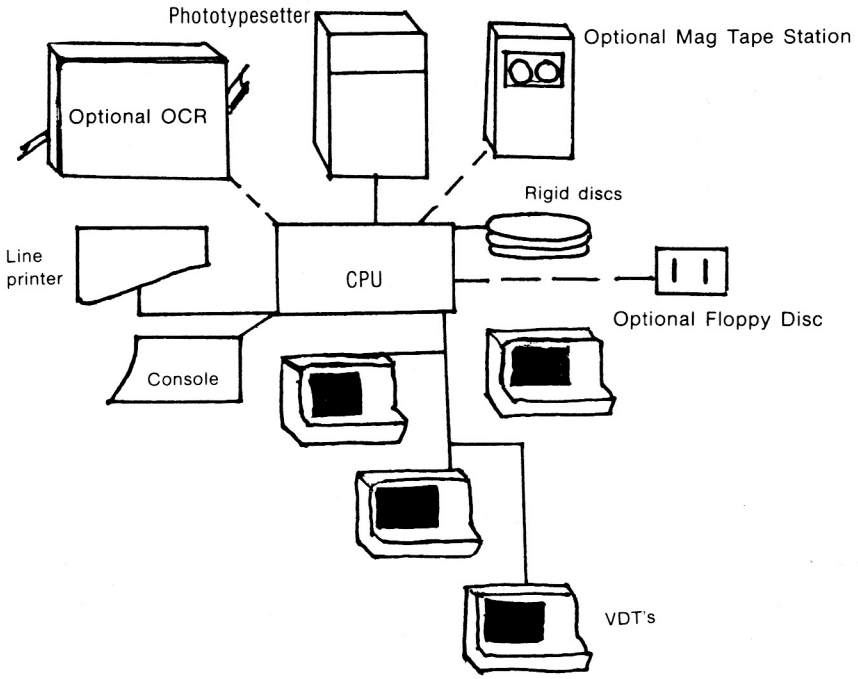
And then there is also the evolution upward from the single terminal to the "cluster" approach, which implies that several people will be performing different tasks simultaneously. Examples of the small cluster approach would include the Telex 2100 (which is as much a word processor as it is a typesetting system, and which drives both a daisy-wheel printer and typesetter.) And the Bobst/Varisystems MTMS is really a string of VariComposers (which are stand-alone devices) linked together by virtue of sharing one common floppy disc drive. Such small systems are generally available in the $30,000 to $50,000 price range, including the outlay for the purchase of one of the less expensive second-generation typesetters. Obviously the price is dependent upon the number of terminals, and whether or not a printer peripheral is desired. Other cost factors would include possible on-line interface of an OCR device, and the degree of complexity involved in supplying an "output driver" for the typesetter. The presumption is that such a typesetter would be operated on-line, and that each of (say) three terminals could queue their text to be dumped to the output typesetter.



Telex 2100 configuration flow chart

IV. Medium-sized configurations

In the range of $50,000 to $100,000 (not including the cost of the typesetter itself) one is likely to find a four-or-more terminal system, possibly with an on-line CRT (cost of the OCR device being extra as well), a line printer for interim output, larger (rigid) disc capacity of from ten "megabytes" (*i.e.*, 10,000,000 characters) to perhaps 40 Mb, and a great deal more sophistication in the file management system. Again the typesetter would be on-line.

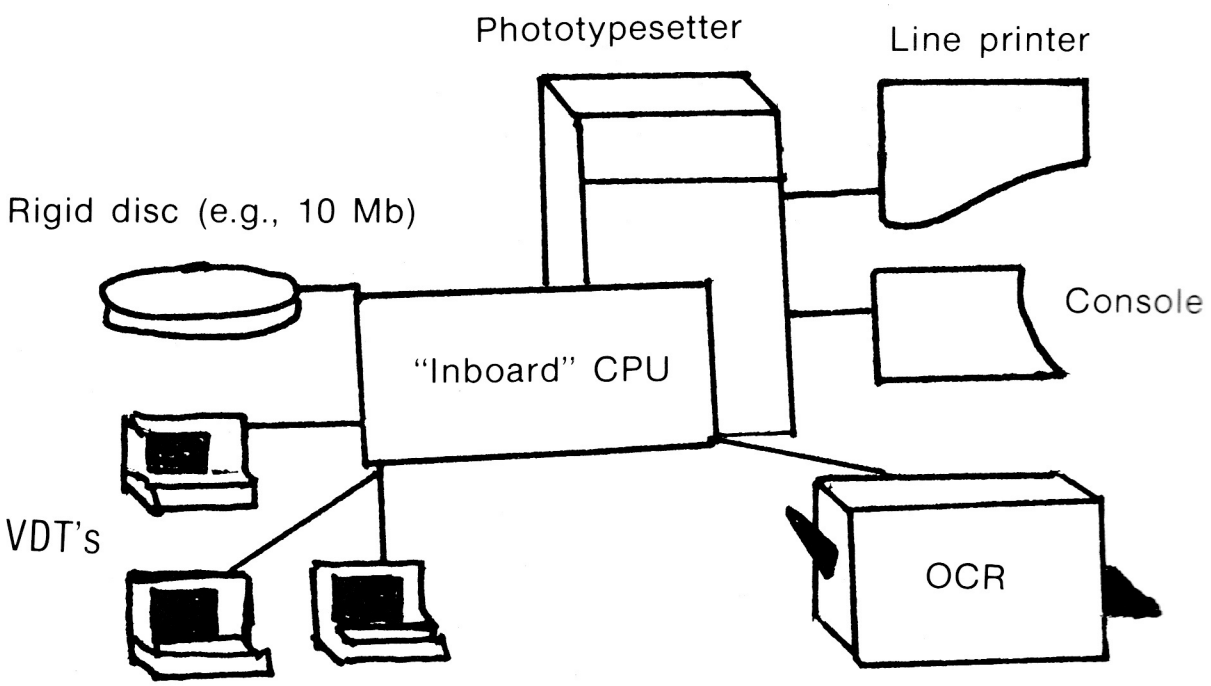


V. An "inboard" typesetting system

At one point compelling arguments were advanced for the use of "in-board" systems. By this it was meant that the same minicomputer which served as the controller for the typesetter (making the end-of-line decisions and also instructing the machine what characters to flash, how much to escape, and when to advance the film) would still have enough time on its hands that it could support several video terminals, even performing h&j and feeding its output back to the video terminals for review and editing prior to typesetting. All these functions would be performed in a multi-tasking environment. To the host cpu/controller would be connected one or more disc drives and even a line printer.

This was, indeed, the architecture for the Linotron 303 CRT typesetter. One obvious drawback to this arrangement is that the user cannot, of course, change output devices. Moreover, in terms of system configuration price there was relatively little difference between the cost of a 303 "system" with typesetter, and the cost of a comparable system on-line to the typesetter of your choice.

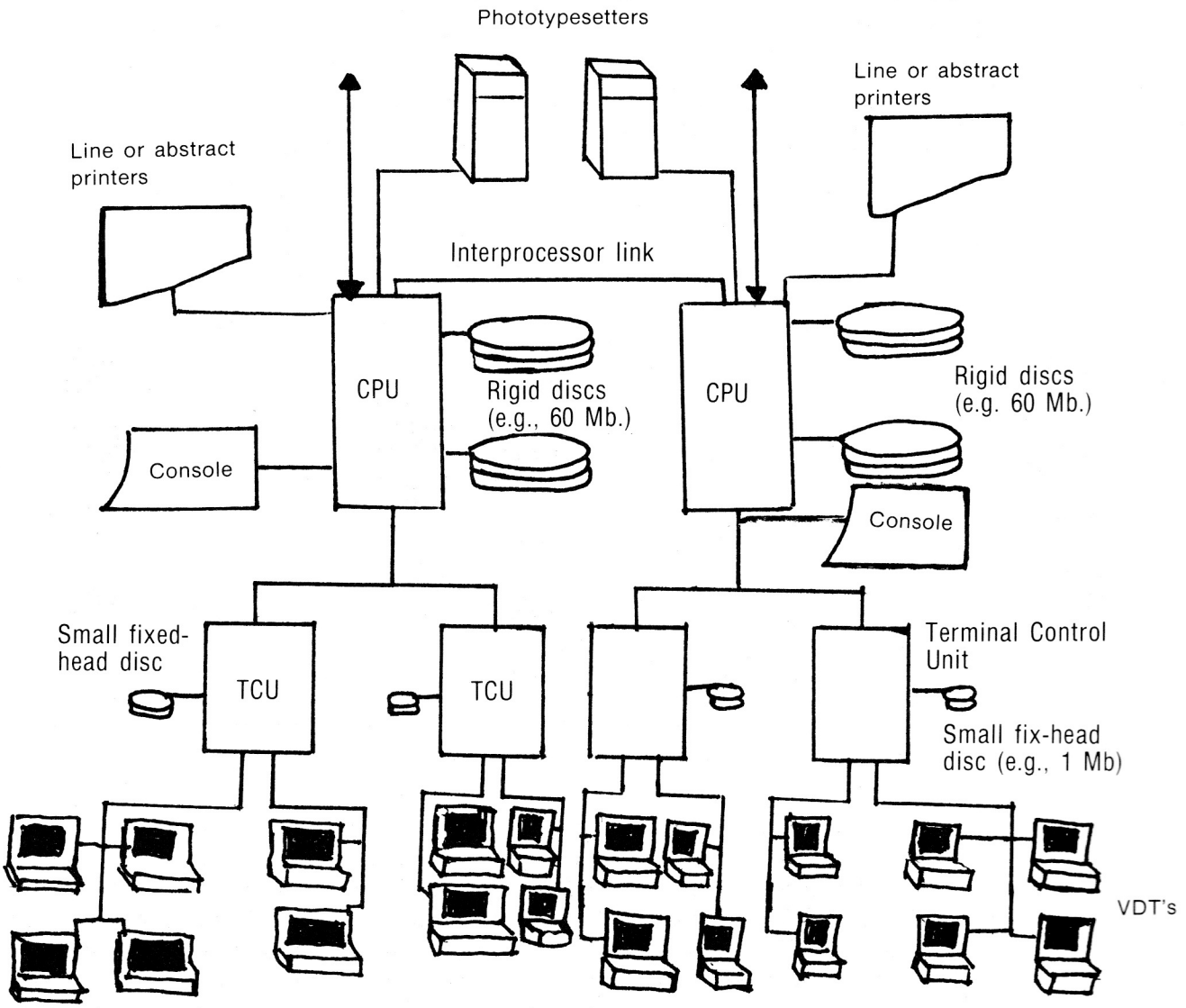
Interestingly, there are some similarities in conception between the "inboard" system and the direct-entry typesetter with peripherals, except that the latter would more likely operate in a foreground/background mode rather than in a true multi-tasking environment.



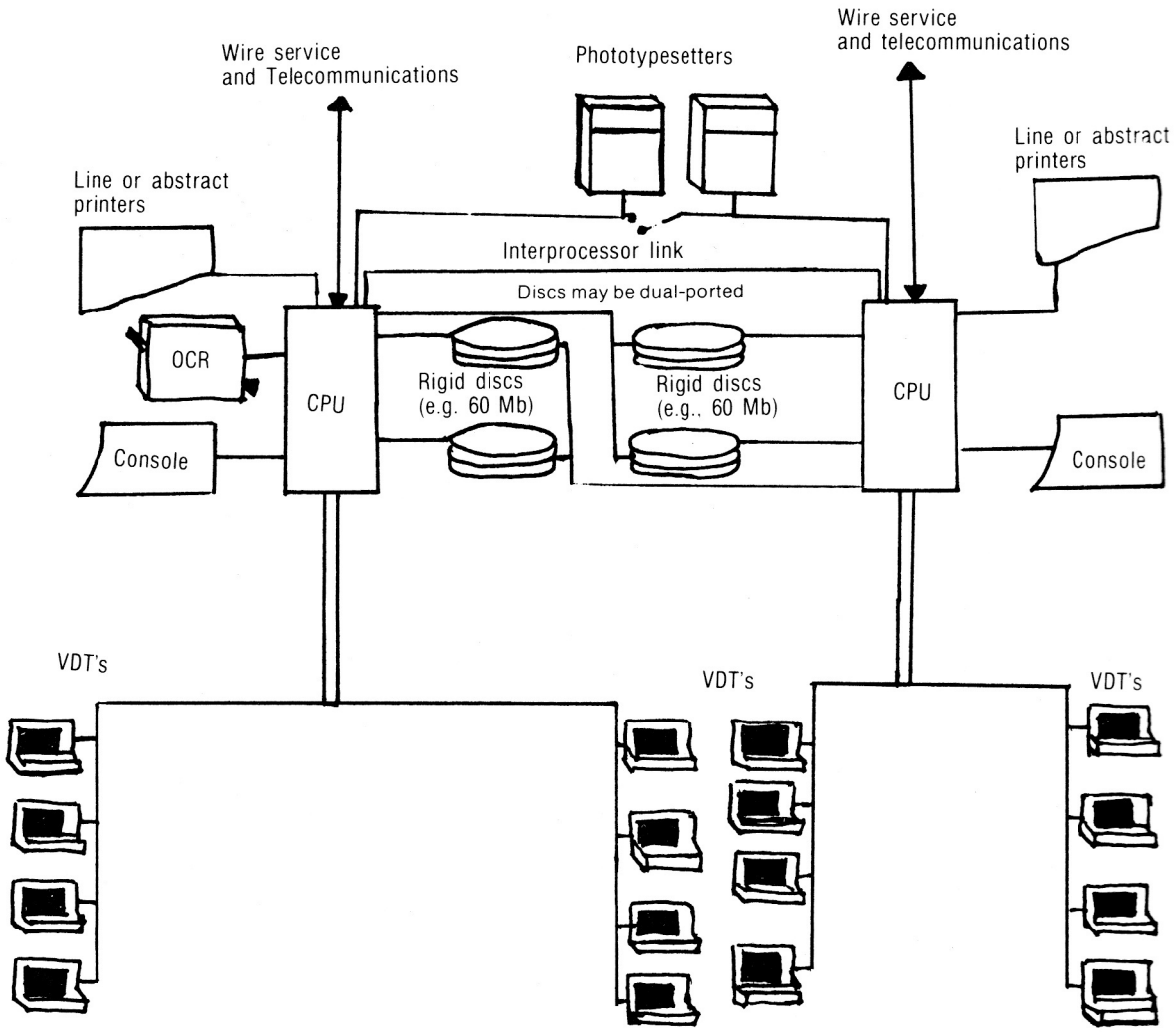
The inboard CPU functions as a typesetter controller, and also services, in a multi-tasking capacity, the demands of the intelligent VDT's, catching OCR input, outputting to the line printer, and serving as a file manager.

VI. Larger system configurations

systems presently selling for well over $100,000 the functions are quite similar to those performed on smaller systems except for the number of terminals supported, the number of peripherals, and the amount of file storage. These systems are common in newspapers, are frequently found in the larger magazine and general publishing establishments, and may also be relevant, if economically viable, for commercial printers and typesetters. Here are two models illustrating different approaches to system architecture:



With terminal control units, intelligence may be located in the TCU. TCU's, if equipped with their own discs, may perform h&j "locally," but host computers handle file management and I/O tasks.



Note: VDT's may be interfaced to CPU via bit serial, parallel, or direct memory access. VDT's may be intelligent, or intelligence may reside in the host computer. Character refresh may be performed by host or within the terminal itself.

To "grow" these systems beyond a certain size, and yet to permit all terminals to share a common data base, some form of interprocessor link or data bus may be used. Usually redundant computers are provided in order to furnish "back-up" for the system. Such back up may be "hot" in the sense that the system itself "knows" when one of its components is unable to respond and automatically switches to another unit, or it may require operator intervention to plug or unplug certain components, either using stand-by equipment, or borrowing components from other similar configurations. For example, in a newspaper situation, a "business" computer might be called upon to perform typesetting functions, or vice versa. Redundancy of equipment, including redundant data bases (so that all files are copied to other disc drive/controller components) will add significantly to the cost of such a system.

Summary

The system choices and configurations described above do not necessarily contemplate the handling of interactive graphics or interactive pagination functions. We leave it to the reader's imagination to incorporate these concepts in whatever manner he sees fit.

It should be evident that "modern composition" embraces a wide number of techniques, systems, concepts, and ways of organizing the intellectual and mechanical processes of "publishing information." As we said at the very beginning, we do not know "for sure" where all of this is leading. The evolution we have witnessed in the past few short years is only a prelude. Those who take the time and trouble to consider this evolution in its full perspective will develop the capability of adopting, adapting, and directing the changes it implies.

We have lived in a world of change, but for many years these events have had very little impact upon the processes of writing, editing, and composing material for the printed page. Now it is our turn. We are caught up in it.

There is no escape, but there is no particular need—nor should there be any desire—to escape. We are confronted with many challenges and opportunities. In terms of technology, we are suffering from an "embarrassment of riches." Let us hope that we make the most of them!

Review Questions:

1. What are the factors which would make it necessary or desirable for a small typesetting "system" to provide some means to record and play back keystrokes?
2. What type of applications might require additional editing flexibility?
3. Enumerate some of the possible system configurations which can be built up around a direct-entry typesetter.
4. What are the various possibilities for system configurations based on the use of an "intelligent" typesetter which would perform hyphenation and justification as well as produce the final output?
5. What is gained when the h&j functions are performed at the terminal level and the "intelligent" typesetter is used as a "slave" device?
6. What is gained when an intelligent OCR device is used to provide input to a system?
7. How are the capabilites of a typesetting system affected when random-access magnetic disc storage is added to the configuration?
8. What are the differences between an "inboard" and an "outboard" system?
9. How might the use of "distributed" intelligence contribute to a system's capabilities?
10. Match each system "model" described with a potential user and set of applications.
11. What production situations might call for "back-up" or redundancy in a system configuration? What are some of the ways in which this could be achieved?

Index

A

AM International
  Comp/Set 500—100, 111, 192, 2(il.) 201, 383
  Comp/Set 4510—198
  Models 707, 744, 748—101
  Scan/Set—*(il.)* 267
AM VariTyper *see* AM International
a.a.'s *(author's alterations)*—190, 193
abridging—381
abstracting—381
accents—253*ff.*
access (sequential or random)—178
access times—310
accuracy rates *(in OCR)*—284
acoustic coupler—326
ad display make-up *see* display ad make-up
addresses *(computer)*—166
Addressograph-Multigraph *see* AM International
Aiken, Howard and Norbert Wiener—159
airline schedules—382
AKI (Automix Keyboards)—257*ff.*
  CIM-80—(il.) 259
  CIM-100—*(il.)* 259
  PCI-120—*(il.)* 259
AKI editing terminals
  PRO series *(il.)*—290
  Ultra series *(il.)*—290
ALGOL—186
algorithm *(for hyphenation)*—226, 230-1
"alpha" characters—9
alphanumeric data—167, 179
Alphatype, AlphaComp—111, 389
AMcomp *(now Logicon TPS/6000)*
  VDT terminal *(il.)*—302
American Newspaper Publishers Association, *see* ANPA/RI
American Type Founders *see* ATF
analog signals—347
Analytical Engine—158
and-gate *(diag.)*—163
ANPA/RI—287
architecture *(computer)*—165
area composition *see* display ad make-up
  See also pagination
arithmetic and logic *(computer)*—166
artwork representation—339
artwork scanning—342*ff.*
ascenders—18
ASCII *("askey")*—173, *(ex.)* 174, 247, 253, 277
assembler program—187
Associated Press—288, 366
Atex—362
  file directory—*(il.)* 316
  keyboard layout—*(il.)* 255
  reporter input terminal—*(il.)* 297
  8000 VDT system— *(il.)* 303, 315
ATF B-8—76—*(il.)* 77
"audit trail"—314
Autologic—137, *(il.)* 365
  Autologic APS-2—141
  Autologic APS-3—141
  Autologic APS-4—143
  Autologic APS-5—143, 365
"Autofunctions"—*(ex.)* 277
automated "cast-off" *see* cast-off
automated page make-up *see* pagination
automated paste-up *see* paste-up
automatic indexing—381
automatic mode—198, 299
Automix Keyboards *see* AKI
"Autostrings"—*(ex.)* 277

B

Babbage, Charles—157
"back-end" systems—318
bar code font—*(ex.)* 275
bar code reader—275
Bardeen, John—160
base alignment—18, 26
BASIC *(program)*—186
"baskets," *see* file management
"batch" corrections, *see* corrections, batch
batch processing—240, 336
baud—64
Baudot code *(Murray)*—64,*ff (il.)*65
Baudot, Emile—64
Bell Laboratories—160
Beloit—324
Benday *(effect)*—354, 356
Berg, N. Edward—302
Bezanson, Howard—227
binary logic *(systems)*—160, 169
"bits"—68, 167*ff.*
Bobst—268
"body" lead—20
book page make-up *see* pagination
book typesetting—190, 328
Boole, George *(Boolean logic)*—160
Booth, A.C.—64
Bowne Time Sharing *("Word One")*—324
Brattain, Walter H.—160
"breaking" text—334
buffer—320
  See also VDT character buffer
Bureau of the Census, *see* United States Bureau of the Census
"burst" mode—165, 179
Byron, Lord—158
byte(s)—167*ff.*, 248, 352

C

camera *(TV type)*—347
Camex—336, *(il.)* 339, 340, *(il.)* 342
"carding"—333-4, 336
Carlson, Jerry A.—191
caster, Monotype—5, 45*ff.*
"cast-off"—335
catalogs—371, 382
cathode ray tube *(CRT)*—120, *(il.)* 292. *See also* CRT
CBS Laboratories—121, 134, 262, 348
channel controller—164
channels *(communicating)*—164
Chapman, Brian—354, 360
character density *(resolution)*—134
character displacement, *see* escapement
character exposure *(second-generation typesetters)*—94*ff.*
character generation *(for VDT's)*—291-3
character height—18
character laydown, *see* escapement
character "mapping"—250
character recognition—275
  See also OCR
character refresh *(in VDT)*—291
character repertoire—248, *(ex.)* 249, 253
  in halftone generation—361
  Monotype—52
  OCR—262*ff.*, 278
  second-generation typesetters—88*ff.*
  typewriting vs. printing—10
  VDT—291
  See also character sets; characters, numbers of
character selection
  second-generation typesetters—88*ff.*
    AM VariTyper 707—101
    CompSet 500 *(diag.)*-100
    Photon 713—88—*(il.)* 89
character set *(typewriter)*—9
  for VDT *(programmable)*—292
character sizing
  in second-generation typesetters—95*ff.*
    Alphatype—95
    AM VariTyper Comp/Set 500—*(diag.)* 100
    Mergenthaler—96
    Photon (Dymo) Pacesetter—107
  in third-generation typesetters—125-6, 136, 138
  See also, lens systems, electronic manipulation
character store—120
  in second-generation typesetters—87*ff*
character widths—1, 6, 18, 109, 200, 204
characters, numbers of—114
classified ads—196, 368
  input by OCR—279
"clocking" *(cassette)*—322
code explosion—318
code structures *(in OCR)*—276
code translators—115; *(in OCR)* 276*ff.*
codes *(typesetting)*—213, 247, 256
coding *(and programming)*—185, 247
Cognitronics—267-268

coherent light source, *see* laser scanning
cold composition—1,2
Collotype—354
Columbia Broadcasting Laboratories, *see* CBS Laboratories
column merge—331
command codes—199*ff.*, 213*ff.*, 256*ff.*, 370
commercial systems *(and OCR)*—268, 390
compiler programs—186, 371
composing stick—*(diag.)* 4, 6
composition, cold—1,2
composition, impact—1,2,7
composition programs—187, 197*ff.*, 371
composition systems—196, 197, 236
Composition Systems Inc.—*(il.)* 303
composition of yellow pages—346
CompStar 190—103, *(diag.)* 104, 109
Compugraphic—102, 132
  Compugraphic 2900—102
  Compugraphic 4900—102
  CompuWriter—102, 108, 389
  OCR reader *(Dest Data)*—268
  EditWriter—389
  Unified Composer—*(il.)* 300
  Videosetter—130, *(diag.)* 131
CompuScan OCR reader—263, 264, 267, 268, 274
  Alpha scanner—*(il.)* 267, 270
  "Free Form III"—279
  Model 370—264
computer composition—317
computer registers—177
computer storage—310
computer typesetting programs—193, 234*ff.*
computer typesetting systems—238
computers—157*ff.*
  speeds—174
  structure—164-5
Computype (CompuEdit)—*(il.)* 302
Context Model 101 OCR reader—*(il.)* 273
continuous tone *(art)*—354
Control Data OCR readers—263
controller *(channel)*—165
copy editing—235
copy fitting—236
copy flow—328
core memory—*(diag.)* 166
core requirements—228
correction routine *(Rocappi)*—240
corrections—192, 237, 239
corrections, "batch" *(with VDT)*—288
corrections *(with OCR)* 279, 283
corrections, random—304
"cottage industry" input—263
counting keyboards, *see* keyboards
counting mechanism
  Monotype—*(il.)* 5,6
  TTS—59*ff.*
Courier font—*(il.)* 274
cpu *(central processing unit or "mainframe")*—166, 308
Crosfield Electronics Magnaset—135
  MagnaScan—350
CRT fonts *(Autologic)*—143
CRT *(digital type)* master sizes—*(il.)*-138
CRT tube—*(diag.)* 120
CRT typesetters—119*ff.*, 196, 347, 365
CRT typesetters *("full face")*—332
CRT typesetters, microfilm attachment—360
cursor controls *(in VDT's)*—287, 291
cursor editing—287, *(il.)* 291, 293, 294*ff.*, 320, 326
cycles *(computer)*—166
Cyphernetics—324

D

"d to a" converter—350
Daconics/Xerox Visual Type System—*(il.)* 325
"Daisy Wheel" printer—320, *(il.)* 321
data bases, 369, 382 *ff.*
data compression—358
data exchange—248
data management—310*ff.*
data manipulation—381
data processing—368ff.
DEC *(Digital Equipment Corp.)*—165
  RK05 disc—312
  RK11 disc controller—312
  VT71/t—*(il.)* 297
declarative instructions—176
"dedicated" programs or systems—240, 369
"define block" command *(VDT)*—301
delete function *(VDT)*—294
delete symbol *(OCR)*—*(ex.)* 280, 281
delimiters—220, 281
DeMorgan, Augustus—162
descenders—18
Diablo printer—320
dictionary *(total)*—232
  See also exception word dictionary
Didot point—22
Difference Engine—158
diffraction theory—359*ff.*
Digiset, *see* Hell Digiset
Digital Equipment Co., *see* DEC
digital representation *(of characters)*—120
digital techniques *(data storage and transmission)*—358
digital-to-analog converter—350
direct-entry typesetter, *see* direct-input typesetter
direct-input typesetter—198, 387, 389
directories *(as data bases)*—373, 382
directories *(file)*—313, 314, *(il.)* 315
disc—*(il.)* 310, 312
  access—310, 312
  controller—310
  "crash"—312
  drive *(il.)* 171
  fixed head—310
disc—*(continued)*
  floppy—313, 321
  memory—168
  moving head—310
  organization—312
  sectors—311
  tracks *(diag.)*—311
disc storage *(of CRT fonts)*—146
disk, *for computer disks see* disc
disk, *(font or film)—88ff.*, 198
diskette, *see* disc, floppy
displacement, *see* escapement
display ad make-up—328
  See also pagination
distortion *(of photographic characters)*—102
  on CRT typesetters—121, 125
DMA *(direct memory access)*—166, 308
documentation of programs—188
Donnelley, R. R.—362
dot matrix *(VDT)* 293
DRUPA—122
Dvorak keyboard—250
Dymo Graphic Systems *(acquired by Itek Composition Division 1979)*
  laser typesetter—356
  Pacesetter—115
  See also, Photon, *and Star Graphic Systems*

E

EBCDIC *("Ebsadick")*—173, *(ex.)* 174, 253, 277
Eckert, J. P. and J. W. Mauchly—159
ECRM *(acquired by AM International in 1979)*
  Autokon 8400 electronic camera—365
  Autoreader—*(il.)* 267
    feed mechanism—269
  Model 4400 OCR—279
  7600 text editing/composition system—*(il.)* 303
editing and corrections—198, 235
  in OCR—279*ff.*
editing by cursor, *see* cursor editing
editing terminals—197, 287*ff.*
  as counting keyboards—299
  in word processing—319*ff.*
  time-shared systems *(line editors)*—324
  See also, VDT's
editorial control *(of copy)*—315, 317
editorial control *(of illustrations)*—346
editorial interface—383
editorial process—236, 319
editorial systems—317, 319, 321, 385
editorial tools—287*ff.*
editorial/typesetting interface—*i*, 189*ff.*
"efficient" mode *(of VDT utilization)*—291
18-unit system—5, 25, 54, 198, 201
electrical circuits—159, 162 *(diags.)* 163
electronic color separation—360

Electronic Numerical Integrator and Calculator, *see* ENIAC
electronic sizing—125, 126, 138
em dash—10
en dash—10
en space—22-3, 256
end-of-line decision—5, 113, 198, 223, 238, 247, 258, 300
end of paragraph—11
ENIAC—159
error rates *(in OCR)*—285
escape codes--256
escapement—1, 17, 22, 24,25, 95*ff.*
(diags.) 99, 100, 101
Comp/Set 500 *(diag.)* 100
Fotosetter—*(diag.)* 74
Monophoto—75
Photon 200—7,8
escapement values, storage of—108
See also, width tables, character widths
exception word dictionary—117, 228
exception words—229
explosion *(of text entries)*—368,381
Extended Binary Coded Decimal Interchange Code, *see* EBCDIC
extraction *(of data)*—381

F

facing pages—333
facsimile transmission—353*ff.*
Farm Journal—191
Farnsworth *(inventor of image dissector tube)*—130
feature analysis *(in OCR)*—275
fiber optics—101*ff.*
face plate—132, 146
fielded data—279
fields, fixed—374*ff.*
variable length—379
file directories, *see* directories, file
file management—198, 313, 376*ff.*
"in baskets"—315
pointers—315
queues—315
files, structured—372
firmware—117, 166
fixed-head disc, *see* disc
fixed space(s)—11, 23, 214, 247, 333
"flags"—378, *(ex.)* 379*ff.*
flexibility, *see* typographic flexibility
flip-flop—*(diag.)* 163
flow charts—180, *(ex.)* 182
flush *(left, right, center)—(ex.)* 12
See also, quad *(left, right, center)*
"flying spot" scanner, *see* scanners
font compression—141
fonts—30, 114, 250
Justowriter-76
Mergenthaler Linofilm—85
Monophoto—85
OCR—*(ex.)* 263, 274, 275, 280, 282; 284
fonts—*(continued)*
second-generation phototypesetters—88
footnotes—334
format(s)—11, 13, 14, 178, 214*ff., (ex.)* 217, 247,
stored—323
"forms control" *(in OCR)*—283
FORTRAN—186
Fototronic (CRT, 1200, 4000, TxT, 7400), *see* Harris Fototronic
foundry type—17
fourth generation typesetters, *see* Dymo laser typesetter, Monotype Lasercomp
fractions—253
Friden Justowriter—76, *(il.)* 77, 109
front-end intelligence, 117, 197, 258
front-end systems—315, 319, 321
full-face character generator *(CRT typesetter)*—134

G

galley composition—329
Gannett newspapers—58
gas plasma *(display)*—325
General Electric—324
Terminet—326
generations *(of typesetters)*—119
"global" search—277
Goldberg, Hiram—262
"golfball" *(typing element)*—1, 248
GPO, *see* United States Government Printing Office
Grafix I scanner, *see* Information International
graphics—251, 253
graphics *(digitized)*—352, 366
Graphic Systems—101, 268
gravure cylinders—358
screens—354
gray scale—355*ff.*, 366
grid *(photographic, for CRT typesetter)*
Videosetter—*(diag.)* 132

H

"H & J," *see* hyphenation and justification
See also, justification
halftone(s)
contour information—359*ff.*
digitizing—361
dots—350, 359
economics of—366
film chemistry and—362
generation—346, 347*ff., (il.)* 350-1, 363
gray scale level—361*ff.*
masking—365
screens—354
storage—366
"windowing"—366
"hangers"—236
hanging indent—*(ex.)* 13
hard copy—288
hardware/software—115, 174
hardwired logic—117, 128, 299
Harris *(formerly Harris Intertype)*
Fotosetter—72*ff.*
Fototronic CRT—142, 147, 365
Fototronic 4000— *(diag.)* 147
Fototronic 7400—147
Intertype Monarch—63
1100 VDT—287,*(il.)* 288
1500 VDT—*(il.)* 306
2200, 2250 ad layout terminals—337-9, *(il.)* 338.
2500 newspaper system—*(il.)* 316, 318
head *(caption)—(ex.)* 13
"header" sheets *(in OCR)*—277, 284
Heil, Dr. -Ing. Rudolf—139
Digigraph 40A20—357
Digiset—135, 139, 357
Helioklischograph—358
Hendrix Electronics
OCR-2 *(il.)* 267, 272, 279
5102 VDT—288
5200 VDT—289
5700 VDT *(keyboard) (il.)* 297
heuristic programs—264
hexadecimal counting—169*ff.*
Higonnet, Rene—72
Hollerith, Herman—158
See also, punched cards
homograph—227-8
Honeywell, model 316 *(computer)*—128
hot metal *(type)* composition *("hot type")*—1, 2, 17, 204
hot metal shop production—190
hyphenation—197, 202, 209, 219, *(chart)* 230-1 253, 256, 258, 370
reprocessable—241
with VDT—300
See also, justification
hyphenation verification—232
See also, exception word dictionary
hyphens—254
"discretionary"—254, 277
"machine"—255

I

IBM *(Advanced Text Management System)*—324
IBM Executive typewriter)1, *(ex.)*1, 108
IBM MTST *(Mag Card Selectric Typewriter)—(il.)*—320
IBM OCR readers—262
IBM Selectric Composer—2, 240
IBM System Development Division—251
IBM System 370—182, 369
IBM 1130 software—212. 213
IBM 1620 composition software—213
"idiot" tape—199

illustrations—346
  resolution of—350
image dissector tube—*(diag.)*130
image quality *(of CRT typesetting)*—127
Imlac 1550 VDT—*(il.)* 307
impact composition—1, 2, 7
  See also, "strike on" composition
imperative instructions—176
"in baskets," *see* file management
indents—13, *(ex.)* 14, 117, 204-5
index, disc—313
Index Medicus—155
indexing—371-3, 381
information interchange—248, 252
Information International, Inc.—139, 154
  Grafix I scanner—*(il.)* 265
  Model 3600 continuous tone scanner—357, 364
  Model 500 *(and 570)*—135, 148, 357
  Videocomp 800-series—364
information retrieval—313, 388
in-house *(or "in-plant")* typesetting—388
"input"—320
input *(into typesetting system)*—245*ff.*
input devices—10, 165
input perfection—237-8
input tape format—116
insertions *(OCR)*—*(ex.)* 282, 295
instructions *(computer)*—175
integrated circuit chips—291
Intelligent Machine Corp.—262
"intelligent" typesetter—258
interactive graphics—341*ff.*, *(il.)* 342
interactive page make-up *see* pagination
interactive systems—208
interface—*i*, 247
  parallel—305
  VDT—304
  word processing—321
interlacing *(for VDT character generation)*—293
International Typographical Union—76
interword spacing—3, 111, 204
Itek, *see* Dymo Graphic Systems
ITU, *see* International Typographical Union

J

Jacquard, Joseph Marie—158
Jessel, Walter—251
justification—3,4,6, *(ex.)* 8, 111*ff.*,*(ex.)* 212, 252, 370
  by hand—4
  hot metal—4
  hyphenless—117, *(ex.)* 219
  in composition
    flow chart—206
    Fotosetter—73*ff.*
    multi line—233
    programs—191*ff.*, *(il.)* 203
    vertical—330, 333
  with logic hyphenation—117
justifying space(s)—10
Justowriter, *see* Friden

K

kerning—*(ex.)* 18, 125, 255
keyboard—9, *(ex.)* 9, 258, 261
  blind—*(il.)* 258, 261
  counting—111, 258, *(il.)* 261, 300
    as editing terminal—299
  editing—*(il.)* 261
  hard copy—258
  layout—*(il.)* 255, *(il.)* 259
  Monotype—5
  shorthand routines—277
  soft copy—*(il.)* 258
  TTS—59*ff.*

L

labor relations *(and OCR)*—283
Lanier Text Editor—*(il.)* 3323
Lanston, Tolbert—23, 158
  See also, Monotype
laser *(for OCR scanning)*—271, 350
laser platemakers—357
laser scanners, *see* scanners
laser "writers"—355
Lasercomp, *see* Monotype
leader(ing)—*(ex.)* 12, 214
leading—19, *(ex.)* 20, 213, 329
  reverse—116, 332
lens system(s)—27
  devices
    AM Comp/Set 500—*(diag.)* 100
    AM Model 707—*(diag.)* 101
    Compugraphic—95
    Fotosetter—73
    Linofilm—86, 96
    Linofilm SuperQuick—*(diag.)* 94
    Monotype—*(diag.)* 75
    Photon Model 200—*(diag.)* 80
    Photon Model 900—89, *(diag.)* 91
    V-I-P—95
  optical properties—362
  rotating—114
  turrets—97, 101, 102, 114
letterpress—72
letterspacing—3, 199, 213
"Lexical-Graphical" *(CRT)* typesetter—348*ff.*, *(il.)* 350-1
Lexitron Videotype—*(il.)* 322
ligatures—10, 17, 213, *(ex.)* 253, 255
line drawings—347
line editors—324
line length *(calculation of)*—199
line length *(maximum in phototypesetters)*—115
linecaster(s)—4, 5, 34, 37, *(il.)* 39
  operation of—34*ff.*
  TTS and—58, 59, 62, 63
  wire service input—58, 59
Linofilm, *see* Mergenthaler
Linotron, *see* Mergenthaler
Linotype—1, *(il.)* 34-9
  Elektron—63
  Intertype Monarch—63
  See also, Harris, Mergenthaler
Linotype-Paul, *see* Mergenthaler
lists *(alignment of)*—11
Lithomat, *see* Photon
lock-up—17
logic, Boolean—160
logos *(logotypes)*—*(ex.)* 18, 346, 357
"look-ahead" logic—145
  of CRT typesetters—147
loose lines—204
Lovelace, Countess of—152
LSI circuits—160

M

machine language—185
machine-readable input—269
MacIntosh, Ronald—121
magazine production—191, 328
magnetic cartridges—321
magnetic cassettes—322
magnetic tape—177, *(il.)* 178, 376
"main frame" *(computer)*—164-5
Manber patent—141
manual *(human)* intervention *(in OCR)*—275
"mapping," *see* character mapping
matrix
  linecaster—4, 5, 37, *(il.)* 39
  matching *(in OCR)*—275
  Monotype—*(il.)* 50
  VDT—*(il.)* 292
Mauchly, J. W., and J. P. Eckert—159
Medlars—155
memory *(computer)*—164, 258
  partition—240
Mergenthaler *(company)*
  CorRecTerm—*(il.)* 288
  Linofilm—82, *ff.*83, 111
  Linotron 202—149*ff.*, *(il.)* 150 *and* 152
  Linotron 303—*(diag.)* 129, 130
  Linotron 404—148, *(il.)* 151
  Linotron 505—121*ff.*, *(diag.)* 123-4
  Linotron 606—148, *(diag.)* 149
  Linotron 1010—121, 134, *(diag. and il.)* 350-1, 370
  MVP editing system—290
  NJ300—*(il.)* 261
  V-I-P—*(diag.)* 103, 105, 114, 115, *(il.)* 150, 257, 290
  Wide range SuperQuick—93, *(diag.)* 94
MGD Metro-reader—*(il.)* 267, 281
  Metro-set—145*ff.*
microcomputers—113, 197
microwave transmission—*(il.)* 362-3
Miehle-Goss-Dexter, *see* MGD
Mill, John Stuart—162
minicomputers—113
mnemonics—257
modem—179
"moire" effect—355

Monotype—1, 5, 6, 24, *(il.)* 46, 158
history—45
keyboard—*(il.)* 46, 49, 55
Lasercomp—154, 356, 357
Monophoto—72, 74, *(diag.)* 75
operation—47
Morey, Walter—58, 63
adaptation of Murray-Baudot code—65
"morgue" *(newspaper)*—383
mortising—*(ex.)* 17
MOS *(metal oxide silicon)* technology—160
"move" function *(in VDT)*—301, 304
moving-head disc—178, 310, *(il.)* 311
multi-column setting—329, 332
multi-processing—310, 369
multi-tasking—310
Murray code, *see* Baudot code
mutt, *see* em
Mycrotek VDT—*(il.)* 307

N

National Geographic Society—254-6
negator—163
newspaper systems—195, 309*ff.*, 328, 383, 385
newspapers and OCR—*266ff.*
Newsweek—238, 343
Nipkow disc—271
"no-break" text blocks—334
non-alpha characters—10
numerals—254
nut, *see* en

O

object programs—176
obliquing—125
OBR *(optical bar code reader)*—*(il)* 267, 268, *(il.)* 275
OCR *(optical character recognition or reader)*—*262ff.*
and video terminals—268, 279, 286
as word processing interface—324
character recognition logic—*275ff.*
copy preparation—324
devices—385, 390
economics of—266, 268, 283-6
implications of—286
input—*(ex)* 263, 273, 274, 275, 278, 280
See also scanning
installations *(number of)*—262, 268
markets for—268
page readers—269
post-recognition logic—269, 279, *(ex.)* 277, 278, 280
preparation of input—284
pre-recognition logic—283
principles of—*268ff.*
type fonts—*(ex.)* 263, 273, 274, 275, 285
octal counting—170
offset—72
Omnitext—288, *(il.)* 289, *(il.)* 293
operating system—187, 198
Optronics International
drum scanner/writer—*(il.) 352, 365*
Pagitron—358
or-gate—(il.) 163
output—320
coding—247
devices—164, 178
driver—241
speeds *(of typesetters)*—113
overstrike *(in VDT)*—291, 293, 294

P

Pacesetter—*see Dymo Pacesetter: see also* Photon
page make-up—*see* pagination
page readers—*see* OCR
pagination—195, 198, 317, 328*ff.*, 386
automatic—335
book—35
economics of—35
interactive—36, 358
parameters for—35-6
VDT editing and—37
"paging" *(VDT)—305, 326*
paper tape—79, 86, 305, 326
code table—65
coding—(il.) 63*ff.*
history—63
merge device—238, 288
punch—*(diag.)* 66
TTS—63, 247
paper transports—*(il.)* 269, 270
parity—168
Pascal, Blaise—157
paste-up—335
Paul, K.S.—121
PDI *(Printing Developments, Inc.)*—360, 366
color scanner—360
"pels," *see* picture elements
peripherals *(computer)*—164
Perry Publications—229, 263
typewriter face for OCR—*(ex.)* 263, 267
phosphor coating *(of CRT tubes)*—120, 136
for VDT—295
photocomposition equipment—385*ff.*
See also chart insertion
See names of individual manufacturers of equipment, such as Autologic, Compugraphic, Dymo, Graphic Systems, Harris, Hell, Information International, Mergenthaler, Monotype, Photon, RCA, Star, *etc.*
photo diode arrays—272, 350
photographic characters *(for halftone generation)*—361
photographic storage *(of 3rd generation typesetter images)*—120*ff.*
Photomultipliers—128
Photon—72, 79
development and operations—78, *ff.*, *(diag.)* 80, *(il.)* 101
Pacesetter—*(diag.)* 107
Model 560—*(il.)* 81, 82
713, 530, 540, 560 *(models)*—110
900 *(Zip)*—89, *(diag.)*, 91, 121
See also Dymo, Itek
phototypesetters—*72ff.*, 278
high-speed—329
See also inserted chart
"pi plaques"—122
picas—19, 21, 22
piclets—109
picture elements—*355ff.*
pincushion distortion—*see* distortion *(of characters)*
"pixels," *see* picture elements
PL-1 *(program)*—180, 371
platemaking—359
plugboards—110, 159
point size—18, *(ex.)* 21, 62, 213
points—19, 21
Didot—22
portability *(of programs)*—186
post-recognition logic *(OCR)*, *see* OCR
precedence codes—331
in TTS—*68ff.*, 247
pre-press activity—*i*
pre-recognition logic—*see* OCR
price lists—382
PRINT '68—122
printed circuits—160
Printext, *(IBM program)*—212
Printing Developments, Inc., *see* PDI *and* Time, Inc.
printing plates, photographic—72
printout, proofreading of—*(ex.)* 240
probability analysis, *(hyphenation)*—224
"production" systems—317
programs, computer—166, 175, 177, 180*ff.*, 186, 203, 206, 207
heuristic—264
typesetting—*(ex.) 181ff.*
data processing/typesetting—*373ff*
See also software
proofreading *(OCR input)*—266
proportional type—2, 6, 8
pseudo-font—250, 252
pseudo-input—241
punched cards—158, *(il.)* 161, 164, 237, 374
Purdy, Peter, and Ronald MacIntosh—121

Q

quad *(left, center, right)*—11, *(ex.)* 12, 208, 215, 256
"qwerty" keyboard—250
queues, *see* file management
Qume printer—320

R

ragged setting—3, *(ex.)* 212
RAM—358
Randolph, Woodruff—76
random access devices—178, 258
raster scan, *see* VDT raster scan
Raytheon—339
  CAM terminal—*(il.)* 345
  Raycomp 100—*(il.)* 341
RC *(resin coated)* paper—329
RCA—262
  VideoComp—139*ff.*
  VideoComp 820—139
  VideoComp 830—140
  See also Information International
read-only memory, *see* ROM
readability *(of type)*—8
real-time systems—310
Recognition Equipment, Inc.—263
recognition logic—272
  omni-font—264
Redactron word processing system—*(il.)* 321
redundancy elimination *(in files)*—381
refreshing *(of VDT characters) see* character refresh
registers *(computer)*—177
REI, *see* Recognition Equipment, Inc.
relative units of measurement—23, 25
remote printing—359, 366
reperforating unit—238
resolution *(of CRT typesetters)*—127
retrieval *(of text)*—313
reverse leading, *see* leading, reverse
ROM—291, 321
rotating mirror *(method of escapement)*—102
routine *(program)*—181
routing *(in file management)*—315
"rubout" function *(for VDT)*—296
Russell, Bertrand—162

S

Scan Data OCR readers—263-4
scanner
  drum—350*ff.*
  flat bed—350*ff.*
  "flying spot"—*(diag.)* 270, 352*ff.*, 363
  laser—*(diag.)* 271
  multipoint—271-2
  vidicon—*(diag.)* 272, 348, 353
scanning—262*ff.*
  logic—276
  mechanism—269, 270
  OCR—262*ff.*, 269
  of art work—348
  of typeset documents—264
Schreiber, Prof. William—360, 366
Schwartz patent—141
screen resolution—358
screening—354*ff.*
  See also halftone generation
scrolling *(VDT)*—291, 301, 304, 342
SEACO *CRT)*—356
search function *(in VDT)*—304
search, global—255
"second-generation" phototypesetters—72*ff.*
  principles—87*ff.*
  See charts inserted in book.
segmented disks—*n.*, 115
selector channels—164
self-scan display—301
semi-conductor, *see* transistor
service bureaus, *see* typesetting service bureaus
set *(size)*—30, 213
set-width tables—204
  See also character widths, eighteen unit systems, unitized fonts
Seybold Report—*n.* 113
shared intelligence, *see* shared logic
Shatzkin, Leonard—193
Shockley, William—160
side bearings—21
signset—252, 255, 260, 276
Singer Graphic Systems—101
  8400 VDT—288
  Justotext—108
  See also Friden
sizes *(number of)*—114
size ranges *(of type)*—115
  in digital CRT typesetters—138*ff.*
sizing, *see* character sizing
skew *(in OCR)*—*(il.)* 274, 283
slaves *(in typesetting)*—111, 116
  Mergenthaler Linofilm—85
small caps—253
soft copy display *(VDT)*—301
solid setting—19, 21
sorting and sequencing—374*ff.*, 383
spacebands—*(diag.)* 4,5, *(Linotype il.)* 38, 111
  See also justification
spacebar—5,6
"space in" and "out"—4
spaces, *see also* em, en, fixed, thin, justifying
speed *(of phototypesetters)*—113
  of CRT typesetters—127
stabilization paper—329
"stand-alone" computer—197
"stand-alone" systems—299, 320
Star Graphic systems—. 103
  See also Dymo (Itek) Graphic Systems
"stenotype"—248
"straddle words"—220
Strauss, Victor—360
"strike-on" composition, *see* impact composition
"strike-outs," *see* delete
strikes *(ITU)*—76
string substitution—178, 256*ff.*
stroke information—135
storage *(computer)*—166*ff.*
stored formats, *see* formats
subject programs—186
supershift—254
"systems," *see* computers, typesetting, editorial, front-end, production

T

tab cards, *see* punched cards
table look-up—171, 201, 255, 375
tabular material—*(ex.)* 15, 216
tabulating machine, *see* electrical tabulating machine
Taft-Hartley Act—76
take length (maximum)—116
Tal-Star *(VDT)*—*(il.)* 307
tape control *(functions under—of typesetters)*—116
tape, magnetic—*(il.)* 178
tape merging—*(il.)* 238, 288
tape *(paper)*—*(il.)* 64, 66, 177
tape "perfecting"—288
Taplin, *see* Context
Tektronix—*(il.)* 345
telephone books—371
thin space—11, 24
third generation typesetters *(CRT)*—119*ff.*, 253
  digital machines—135*ff.*
  photographic masters—122*ff.*
  See also CRT typesetters
throughput speed—228
Time, Inc.—238*ff.*, 343, 360
time sharing—310, 324, 369
"track ball" *(as cursor control)*—294
transistor, *(advent of)*—160
translate tables—249
translators, *see* code translators
transports, paper or document—269
triclets—109
Troxel, Donald—366
TTS *(for teletypesetter)*—58
  coding—60*ff.*,276
  keyboard—60, 247
  illus.—60-62
  multiface perforator—60
  R. R. Donnelley and—59
  standard perforator *(operation of)*—59
  Time, Inc., *(use of)*—59
  unit count fonts—59
TV raster scan *(for VDT)*—*(il.)*) 292, 353*ff.*
type
  advantages of—370
  characteristics of—17
design—17, 21, 26, 30*ff.*
  Monotype—56
  faces
    classification—27-8
    Monotype—53
  forms—2, *(diag.)* 3
  foundry—17
  hot metal—1, 2, 17
  proportional—2, 6, 8, 370
  sizes—*(ex.)* 21, 25, 26 *(photocomp)*
    See also points, character sizing

typesetting
codes—375, 379
commands—213*ff.*, 278
conventions—248
printers'—9
service bureaus—369, 371, 388
speeds—113
typesetting systems—189*ff.*, 193, 385*ff.*
typewriter—1*ff.*
impact on typesetting—256
input for OCR-262ff.
typewriters, automated, *see* word processing
typing element—1
typographic flexibility—114

U

unibus—*(diag.)* 166
uniform widths—1
unit-count systems
Justowriter—76
Photon—79
TTS—59*ff.*
unit system, *see* escapement
See also eighteen unit system
United States Bureau of the Census—158
United States Government Printing Office *(GPO)*—134, 348, 370
United States of American Standard Code for Information Interchange, *(USASCII) see* ASCII
unitized fonts—109
upper and lower case conversion—379
"upstream" *(in copy flow process)*—319
USACII *("yousaskey"), see* ASCII
U.S. News & World Report—238, *(system il.)* 363

V

vacuum tube technology—160
Varisystems, Inc. *(now Bobst Graphic Systems)*
Varicomposer 2000—*(il.)* 261
Varicomposer—*(il.)* 297, 300, 314
VDT *(or VDU)*—*(il.)* 177, 195, 287*ff.*
as on "on-line" terminal—304
character buffer—291, 296, 304
character image quality—291
character set, *see also* charaacter repertoire
diagrams—292
editing features—293ff.
VDT-CPU relationship—304ff.
vectors—*(used for VDT character imaging) —n.* 291, 307
verification *(punched card)*—237
video display terminal, *see* VDT
video terminal systems—279, 308, 362
architecture—*(diag.)* 298
clusters—*(diag.)* 298, 299
VideoComp, *see* Information International
See also RCA
vidicon *(OCR)* scanners, *see* scanners
"virtual" systems—369
visual display *(keyboards)*—258
vonNeuman, John—160
vowel group analysis—222
Vydec Text Editor—323

W

Waverly Press—227
Westar satellite—364
Wheatstone, Prof. Charles—63
white space—247
Whitehead, Alfred North—162
width tables, *see* width values
width values—111, 253, 258
calculation of—201*ff.*
storage of
by plugboard—109
in memory—110
on film disk—111
"widows"—236, 332
Wiener, Norbert and Howard H. Aiken—159
wire wrapping—1160
window *(in CRT typesetting)*—143
wire service input—58-9, 117
word division, *see* hyphenation
word processing—*i*, 319*ff.*
word strings—222
word wrap—*(il.)* 293, 294
words *(computer)*—168
Wright-Patterson Army Air Force Base—134, 348
writer responsibility *(in OCR)*—284
writers and the writing process—*see Introduction*, 234
"Wylbur"—324

X, Y, Z

"X" height—19
xenon flash—101
Xerox *(Daconics)* Visual Type System *(il.)*—325
yellow pages composition, *see* composition
"z" height—19
"zoom" lens system—27, 96, 97
Comp/Set 500—100
Linofilm—85

List of Illustrations and Diagrams

Illustrations are listed alphabetically according to the subject matter described or the context in which they appear.

ATF-B-8 recorder and reproducer units—77
Character height—18
Character matrices—292
Character sets *(Greek, math,, logic)*—249
Code structures
  Morey (TTS) adaptations of Murray-Baudot coding—65
  ASACII and EBCDIC coding—174
Computer concepts: and-gate—163
  or-gate—163
  flip-flop—163
Corrections, merging—243-4
Corrections to computer file—240
CRT, cross section *(diag.)*—292
  digitized form of characters—137
  functional diagram—120
  principles of character generation—131
CRT typesetters
  Autologic digitized characters—137
  Compugraphic Videosetter, photo optics—131
  Compugraphic Videosetter photographic grid store—132
  Linotron 202 digitized characters *(enlarged)*—150
  Linotron 202 schematic—152
  Linotron 303—photo optics—129
  Linotron 404 digitized characters *(enlarged)*—151
  Linotron 404 schematic—152
  Linotron 505 front-end—123
  Linotron 505 back-end—124
  Linotron 606 digitized characters *(enlarged)*—151
  Linotron 606 schematic—149
Cursor control keys (Omntext 1500)—291
Directories:
  Atex 8000 queue—315
  Harris 2500 system short—316
  Harris 2500 long—316
  Omnitext 1500 job—308
Display ad composition:
  Camex 135 "picking" parameters—340
  Camex 135 proof printer output—340
    contrasted with typeset output—340
Harris 2200 terminals—338
  ad copy on screen—338
Raytheon's Raycomp-100—341
  grocery ad make-up at terminal—341
Em space—23
Escapement, Fotosetter's principle—74
Floppy discs—312
Floppy disc being inserted in disc drive—312
Flowcharting routines—206
  overflow—207
Flowcharting worksheet—182
Flow charts, algorithm for hyphenation 230-1
Flow charts, program for typesetting—184
Font storage
  Compugraphic 2961 font drum—90
  Compugraphic type grid—132
  Linofilm grid—83
    grid basket—83
    font selection—85
    keyboard—84
  Linofilm Quick grid—90
Foreign language accents—254
Halftone generation
  Information International's continuous tone scanner—363
  Linotron 1010 *(diag.)*—350
    halftone generated on—351
    line drawings output on—350
  Optronics scanner plotter—352
  Photographic characters (50% dots) —361
  U.S. News configuration—363
  U.S. News satellite relay—363
  U.S. News sample page—365
Hot metal printers chase—3
IBM (or Hollerith) card—161
Justification, hand-set type—4
Justowriter recorder & reproducer units—77
Keyboard layouts
  AKI CIM-80, CIM-100, CIT-70, and PCI-120—259
  Atex reporter input—297
  Atex, for National Geographic—255
  DEC VT 71/t—297
  Harris 1500—306
  Hendrix 5700—297
  Varisystems' Varicomposer—297
Keyboards
  G.O Graphics' Memory Text non-counting—261
  Mergenthaler Micomp counting—261
  Mergenthaler NJ300, non-counting—261
  Varisystems' Varicomp 2000 blind noncounting—261
Leading, typesetting columns with different leading—331
Linotype Model 30—38
Linotype components:
  spaceband release mechanism—38
  assembled matrices and spacebands—39
  assembling mechanism diagram—39
  matrices—37
Magnetic tape deck—178
Monotype—46
  arrangement of punches—51
  caster—50
  counting mechanism—49
  index plate—51
  justifying scale—51
  keyboard/mat case layout—55
  matrix case—50 & 51
  matrix comb—50
  normal wedge—51
  paper ribbon—50
  type (set) samples—54
Moving head disc on disc drive (multiple platter)—311
  tracks & sectors *(diag.)*—311
Moving head disc drive—178
OCR *(Optical Character Recognition)*
  Models:
    CompuScan Alpha, ECRM Autoreader, MGD Metro-reader, Hendrix OCR-2, Datatype Scan/Set—267
accessing typographic command codes
    CompuScan, ECRM, MGD—278
  Autostrings and Autofunctions—277
  bar code (Datatype's DF-2)—275
  Cognitronics scanning principles—271
  Context scanning principles—273
  ECRM paper feed mechanism—269
  ECRM vidicon scanner from early series 272
  editing: delete symbols—280
    insertions, vertical crossouts—281-2
  flying spot scanner *(diag.)*—270
  fonts, samples of scannable:
    IBM Courier—274
    CompuScan Perry—280
    ECRM Courier—280
    OCR A (MGD and Graphic Systems)—282
    OCR A, B, Perry & Courier—285
  Information International's GRAFIX I automated data input system—265
    document read by—265

Laser scanner, MGD Metro-reader *(diag.)* 271
paper transport (ECRM)—269
Page make-up (and area composition)
AM 4800 display layout terminal—346
Camex 135—340
Compugraphic's AdVantage—342
Compugraphic Pre-View soft copy representation—345
Harris 2200—338
Hendrix' news layout system—344
Raytheon's Raycomp-100—341
Raytheon experimental demonstration —345
Tektronix storage tube *(use of)*—345
Xenotron's VideoComposer—342
Paper tape—64, 66
PDP-11 architecture—166
Phototypesetters, second generation
AM 707 optical principles—101
AM 725 schematic—99
Comp/Set 500 *(diag.)*—100
CompStar optical system—104
deflected image—105
Dymo (Itek) Pacesetter principles—80
field flattener lens—105
Graphic Systems' (Wang) C/A/T, optical and character laydown principles—102
Monophoto optical system—75
Photon 200 optical principles—80
Photon 200 Admaster—81
Photon 560 Display Master—81
Photon 560 keyboard—82
Photon 713 interior—89
Photon 713 revolving drum—90
Photon "Zip" (900 series) principles—91
V-I-P optical system—103
Raster diagram—359
Raster scan *(TV)*—292
System models:
Direct entry—389-390
Stand-alone typesetters—390-391
Small systems (Telex)—393
Medium systems—393
Inboard—394
Larger scale systems—395-6
Teletypesetting
counting keyboard—62
counting magazine—61
justification—61
perforator—60
spaceband code blade—61
Terminal displays
AKI Automix Keyboards Ultra series —290
AKI Pro series—290
Atex—303
Character matrices—292
Compugraphic Unified Composer—300
CRT deflection circuitry—292
CRT raster scan—292
ECRM 7600—302
Harris 1100—288
Harris 1500 terminal—306
Harris *(Computype)* CompuEdit—302
Hendrix 5200—289
Imlac's Composer 1550—302
Keyboards for terminals—297
Logicon TPS/6000 *(formerly Amcomp)* —302
Mergenthaler CorRecTerm—288
Mergenthaler MVP—290
Omnitext 1500—289
Omnitext cursor key pad—291
Omnitext screen—293
One Systems—294, 307
Syntext—302
TalStar's dual screen terminal—307
Teleram P-1800—307
Terminal architecture—*(diag.) 296*
Varisystems *(Bobst)* Varicomposer —300
Type faces, classification—27-8
VDT/CPU relationship, stand-alone and cluster *(diag.)* 298
Video display terminal—177
Vidicon scanner —*(diag.)* 272
Width card (Linofilm)—85
Width cards, Compugraphic—110
Word processing devices
Daconics/Xerox Visual Type System —325
Daisy wheel printer—321
G.O. Graphics Universal Word Processing Interface—325
IBM Mag Card Selectric typewriter —320
Lanier Text Editor—323
Lexitron's Videotype—322
Redactron System—321
Vydec's Text Editor—323
Wang Word Processor—325
Word wrap
One Systems display before wrap—294
One Systems display after wrap—294